Semantic-based web services for devices selection

Gérald Rocher, Jean-Yves Tigli

LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES DE SOPHIA ANTIPOLIS

UMR7271

SEMANTIC-BASED WEB SERVICES FOR DEVICES SELECTION

Gérald Rocher, Jean-Yves Tigli

EQUIPE **SPARKS**

Rapport de Recherche

09-2015

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) - UMR7271 - UNS CNRS 2000, route des Lucioles - Les Algorithmes - bât. Euclide B 06900 Sophia Antipolis - France http://www.i3s.unice.fr

Membre de UNIVERSITÉ CÔTE D'AZUR

SEMANTIC-BASED WEB SERVICES FOR DEVICES SELECTION

Gérald Rocher[1], Jean-Yves Tigli[2]

Equipe SPARKS

09-2015 - 73 pages

Abstract:

The last decade achievements in computer hardware miniaturization and power consumption reduction has permitted the multiplication of connected devices integrated in everyday life physical objects (chair, table, lamp, etc…) and physical environments (house, building, vehicle, etc…). These devices implement resources interacting with objects (actuator) and/or gathering data (sensor) about themselves, the objects or the environment. Access to these resources is achieved through services exposing their interfaces and allowing communication with the digital world. Widely deployed in so called ambient environments, these devices and services are selected by ambient applications (service matchmaking) that make them work in concert to assist users in several distinct domains (healthcare, smart houses, etc…). This cooperation requires a strong interoperability between devices, firstly achieved by allowing them to communicate. Although work on communication protocols (IoT, Internet of Things) tries to provide a solution to the technological heterogeneity issue, it is still challenging due to the large number of initiatives in this field. Among all the possible solutions, web-services based approach (WoT, Web of Things) is now widely accepted.

With this hypothesis, and going a step further, the purpose of the present document is to address the semantic heterogeneity issue inherent to the large number of heterogeneous devices and services present in the environment targeting multiple domains (smart homes, smart cities, building automation, healthcare, etc…). This heterogeneity is problematic for ambient applications to select, among all the available devices and services, the most relevant ones. We investigate in this document how semantic web technologies can be leveraged to enrich devices and services with semantic annotations used to qualify it (SWoT, Semantic Web of Things) and help applications selection mechanism to increase the relevancy of the selected devices and services.

Key-words: internet of things, semantic web of things

[1] Laboratoire I3S – Université Nice Sophia Antipolis – gerald.rocher@etu.unice.fr

[2] Laboratoire I3S – Université Nice Sophia Antipolis, Polytech’Nice Sophia – jean-yves.tigli @unice.fr

Table of Contents

Table of figures

List of Tables

Publications

[1] G.Rocher, J.Y. Tigli, Stephane Lavirotte, Rahma Daikhi. **Run-Time knowledge model enrichment in SWoT**: A step toward ambient services selection relevancy. 2015 IoT international conference, Seoul.

[2] … and more to come ☺

Introduction

1.1 Motivations

The last decade achievements in computer hardware miniaturization and power consumption reduction has permitted the multiplication of connected devices integrated in everyday life physical objects (chair, table, lamp, etc…) and physical environments (house, building, vehicle, etc…). These devices implement resources interacting with objects (actuator) and/or gathering data (sensor) about themselves, the objects or the environment [1]. Access to these resources is achieved through services exposing their interfaces and allowing communication with the digital world. Widely deployed in so called ambient environments [2], these devices and services are *selected* by ambient applications (service matchmaking) that make them work in concert to assist users in several distinct domains (healthcare, smart houses, etc…). This *cooperation* requires a strong *interoperability* between devices, firstly achieved by allowing them to communicate. Although work on communication protocols (IoT, Internet of Things) tries to provide a solution to the *technological heterogeneity* issue, it is still challenging due to the large number of initiatives in this field [3]. Among all the possible solutions, web-services based approach (WoT, Web of Things) is now widely accepted [4].

With this hypothesis, and going a step further, the purpose of the present document is to address the *semantic heterogeneity* issue inherent to the large number of heterogeneous devices and services present in the environment targeting multiple domains (smart homes, smart cities, building automation, healthcare, etc…)(Figure 1). This heterogeneity is problematic for ambient applications to select, among all the available devices and services, the most *relevant* ones. We investigate in this document how semantic web technologies can be leveraged to enrich devices and services with semantic annotations used to qualify it (SWoT, Semantic Web of Things) and help applications selection mechanism to increase the *relevancy* of the selected devices and services.

1.2 Context

The CONTINUUM[1] project addresses the problem of the service continuity in the context of ambient environments. It aims at defining theoretical models needed to enable software's to dynamically adapt themselves to their environment and ensure the service continuity to mobile users in heterogeneous and physical environments where available resources are variable over time. Three challenges have been identified: (1) the management and adaptation to the context, (2) the management of the semantic heterogeneity and (3) the control of the balance in between the system autonomy and the human control. This project has resulted so far in the development of a middleware (WComp)[5] based on component and services assembly [6] giving the software's the capability to adapt themselves to the multiple context aspects (device interconnection, physical environment state, users, etc…) thereby ensuring service continuity. This document represents a straight continuation of this project and addresses the semantic heterogeneity challenge.

1.3 Structure of the document

In chapter 2 (Knowledge Modeling), we review the main technologies used in the domain of the semantic web to formally describe the knowledge (Paragraph 2.2). We describe the underlying theoretical foundations of these technologies and address it from two points of views: (1) The knowledge expressivity they provide, (2) their computational complexity. Then, we give an overview of the knowledge types to be described in the context of ambient environments (Paragraph. 2.3) and propose a knowledge modeling approach based on heterogeneous devices and services description models (Paragraph. 2.6).

In chapter 3 (Knowledge Extraction), we address the challenge of developing knowledge models and propose some approaches aimed at leveraging the world wide web available formal knowledge and translating it to reusable knowledge models (Paragraph. 3.2). Then we review several standards used to convey the knowledge from the devices to the ambient systems and challenge them with regards to their ability at conveying all the necessary knowledge types to be described (Paragraph. 3.3).

In chapter 4 (Knowledge Integration), we first describe the knowledge base foundations (Paragraph. 4.2) and reasoning capabilities (Paragraph. 4.3). We review: (1) the challenges at dynamically integrating heterogeneous knowledge models in a knowledge base over time (enrichment), (2) the approaches helping to cope with it (Paragraph 4.5) and (3) explain (Paragraph 4.6) the main advantages of having a continuous knowledge enrichment. We then present an overall integration model (Paragraph. 4.7) and validate it on two motivating scenarios (Paragraph. 4.8).

In chapter 5 (Knowledge Management), we review some knowledge base management solutions aimed at: (1) limiting the size of the knowledge base over time (Paragraph. 5.2), and (2) ensuring the quality of the knowledge base content (consistency, coherency and validity)(Paragraph. 5.3).

In chapter 6 (Knowledge Extraction), we give an overview of SPARQL (Paragraph. 6.2), the standard for querying and retrieving information from a knowledge base, and its foundations. We demonstrate that the SPARQL underlying knowledge retrieval model is not suitable when used to execute expressive selection rules on unknown knowledge base content. Therefore, we propose an alternative (Paragraph. 6.3) that is validated on a use-case (Paragraph. 6.4).

The **chapter 7 (Conclusion and Future Work)** finally concludes this document with a short summary and provide an outlook on the potential future work.

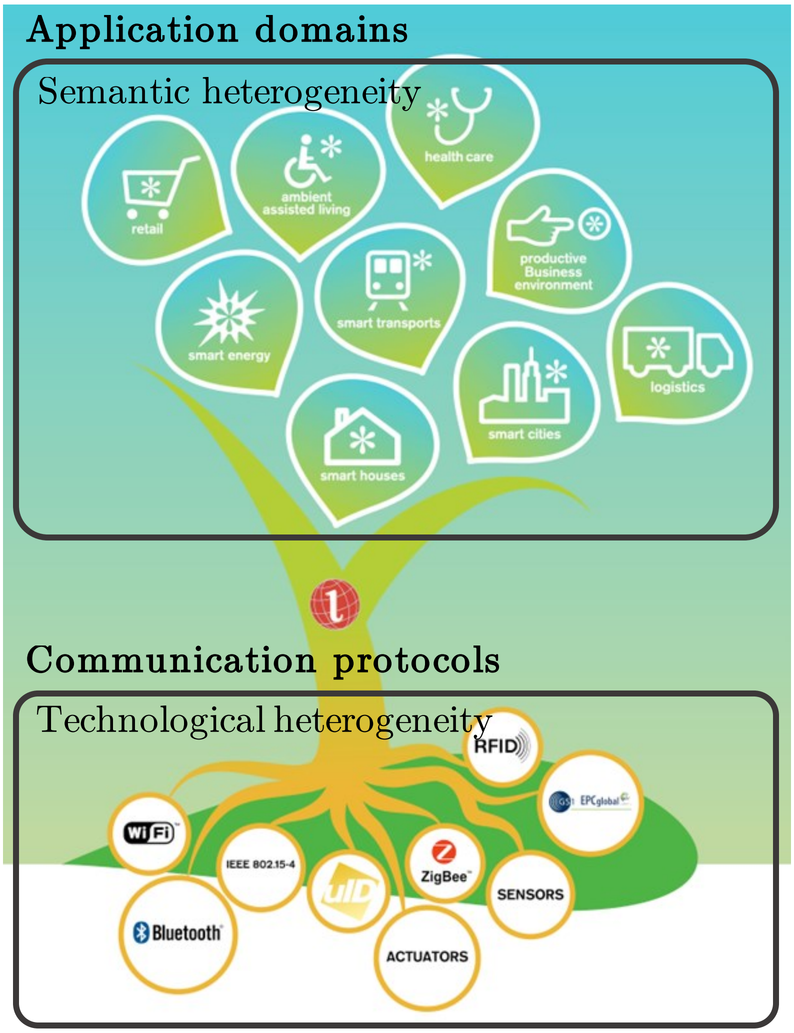


Figure 1 : IoT : Technological & Semantic heterogeneity[1]

[1] Project for service continuity in ubiquitous and mobile computing - French national research agency - ANR-08-VERS-0005.

[1] Internet of Things Architecture - IoT-A

2. Knowledge Modeling

2.1 Introduction

In ambient systems, software applications have to *make decisions* about devices and services to be used in concert to assist users. "Make a decision" is a behavior related to *intelligence* whose foundation is *knowledge* [1]. In the context of ambient software applications, this intelligent behavior has to be achieved *autonomously* by computational means. Indeed, the physical environment the ambient applications interact with is highly dynamic and devices and services availability cannot be anticipated. This concern is well addressed by artificial intelligence (AI) systems based on *knowledge representation models* and *reasoning* techniques. Knowledge representation models are based on *formal* symbols used to represent a set of propositions (concepts, relationships between concepts, etc…) *believed* by the knowledge model designer. Reasoning is the formal manipulation of the symbols to *infer* the representation of a new set of propositions.

As devices communication protocol relies on web-services (WoT, Web of Things), we first investigate in this chapter, what are the technologies used by the Web community to represent and reason on the knowledge available on the web (Semantic Web). Such technologies are, for example, used by web search engines to retrieve and rank the most relevant web information based on user needs or preferences from *metadata* used to qualify web pages content. The problem statement the semantic web technologies help to solve, is close to the problem of selecting, among all the devices and services available in the environment, the most relevant ones to assist users based on their needs, preferences, etc… This has already been well understood and addressed by the WoT community and semantically enriched metadata are now widely used to qualify devices and services (SWoT, Semantic Web of Things).

However, in most of the current work, metadata relies on *ad-hoc* knowledge representation models structuring all the concepts and relationships for a specific domain targeting specific applications (smart homes, smart cities, building automation, healthcare, etc…). Thus, extending the scope of use of the information to multiple applicative domains implies to develop a comprehensive knowledge representation model from heterogeneous knowledge representation models which is unlikely to happen in the SWoT context where domains to cover are countless. In addition, most of the existing domain knowledge description models doesn't follow the semantic web best practices[1], limiting, *de facto*, the reusability of their information outside their initial scope [2].

Then, after having reviewed the several knowledge representation modeling approaches and, after having defined the knowledge to be represented in the context of ambient systems, we propose an approach based on heterogeneous knowledge representation models to qualify and self-describe devices and services through semantically enriched metadata. Unlike current approaches, knowledge description models are restricted to the devices they describe and are embedded in the metadata.

[1] www.w3.org/2014/02/wot/papers/gyrard-2.pdf

2.2 Semantic Web Knowledge Modeling Technologies in a Nutshell

The term “Semantic Web” represents an evolution of the world wide web from a web of interlinked documents targeted towards human consumers to a web of data that makes web content more machine processable (aka Web 3.0).

“I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web — the content, links, and transactions between people and computers. A “Semantic Web”, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The “intelligent agents” people have touted for ages will finally materialize.”

— Tim Berners-Lee, *Weaving the Web* [3]

Semantic web technologies are summarized in the “Semantic Web Stack” (Figure 2). There are many good introductory documents available on the semantic web and associated technologies. The reader interested in acquiring a more detailed knowledge in this area will read the following documents: “Semantic Web for the Working Ontologist” [4], “A Semantic Web Primer” [5], “Programming the Semantic Web” [6], etc…

Semantic modeling is the process of *interpreting* the world (or specific domains of interest) by asserting statements about the world, in other words, enouncing constraints on the possible world states. For that purpose, the interesting technologies in the Semantic web stack are (1) triples and URI (Uniform Resource Identifier), (2) RDF (Resource Description Framework) and RDFS (RDF-Schema), (3) OWL (Web Ontology Language), succinctly described in the next paragraphs.

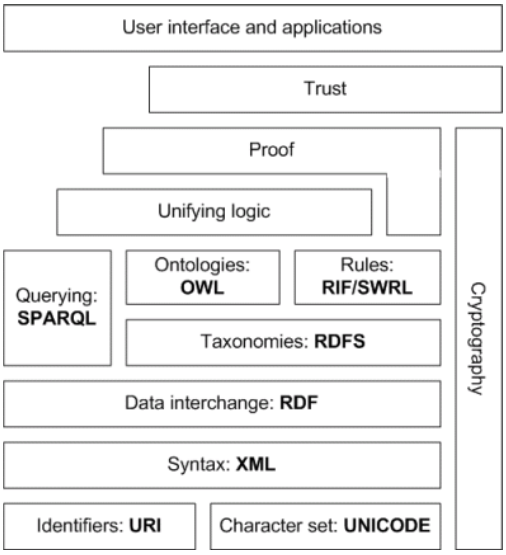


Figure 2 : The Semantic Web stack

Each language brings different *expressivity* capabilities (different abilities to describe the world and express statements about it). The higher is the expressivity the lower is the abstraction of the described world and better could be the relevancy of the selected services.

2.2.1 Triples and URIs

A *triple* describes a *proposition* (a fact, a statement) composed of a *subject*, a *predicate* and an *object*. For example the triple “`iQ700 hasManufacturer Siemens`” is a proposition stating that iQ700 entity is manufactured by Siemens. The predicate, in that case, defines a *relationship* between the subject and the object (the term “Semantics” refers to this simple data model linking together two entities by a relationship). Subject and predicate may also be referred to as *property* and *value* respectively. For example the triple “`iQ700 hasTemperature 125`” is a proposition stating that the entity property “`hasTemperature`” has value “`125`”.

A triple provides an explicit semantics based on agreed *terms*. The terms used in the propositions (subject, predicate, object, property, value type) have to be *unambiguously* defined. For that purpose URIs (Uniform Resource Identifier) [8] are used to uniquely identify each term of a proposition. For example, the statement “`iQ700 hasManufacturer Siemens`” can then be rewritten as follow:

```
<http://www.example.org/iQ700><http://www.example.org/hasManufacturer><http://www.example.
org/Siemens>
```

Therefore, from a semantic standpoint, the terms `<http://www.example.org/iQ700>` and `<http://www.my-ontology.com/iQ700>` **are two different *resources*.**
`<http://www.example.org/>` and `<http://www.my-ontology.com/>` are the *URI prefix*.

2.2.2 RDF

The stack above triples and URI is the Resource Description Framework (RDF). RDF is a graph model used to describe web *resources.* The graph is a set of triples where subject and object are nodes of the graph, and the predicate is a directed edge describing the relationship between nodes (Figure 3). All nodes and edges are defined by URIs except when the edge defines is a property in which case the object is a literal value (plain or typed). RDF introduces a mechanism to make URIs more readable by aliasing URI prefix with an *RDF prefix namespace* (For example “rdf” for `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`).

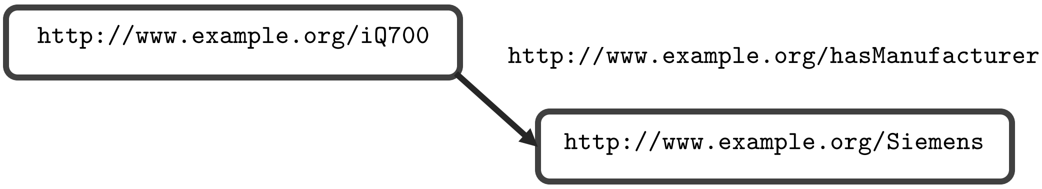

http://www.example.org/iQ700
http://www.example.org/hasManufacturer
http://www.example.org/Siemens

Figure 3 : Simple RDF graph example

While graph representation is good for humans understanding, it is not adequate for machines to process the information it describes. For that reason RDF graph statements are *serialized* to an XML-based syntax document (RDF/XML format). Note that several other serialization formats exist, among which the most used are Turtle (Terse RDF Triple Language) and N3 (N-Triple). In this document, all listings use the abbreviated RDF/XML notation.

An RDF/XML document is represented by the tag `rdf:RDF`. The content of the document is a set of descriptions represented by the tag `rdf:Description`. Each description is a statement about a resource defined either by attribute `rdf:ID` or `rdf:about`. In a general form, the triple `<subject,predicate,object>` can be serialized in the XML/RDF form as follow:

Therefore, the graph in Figure 3 can be serialized in RDF/XML format as follow:

RDF introduces a small *vocabulary* to express statements about the knowledge. For example, `rdf:type` expresses membership hierarchy used to structure the graph. Hence, the graph in Figure 3 can, for example, be structured as shown in Figure 4 (considering two additional statements).

Beside `rdf:type`, RDF defines a vocabulary to write collections (`rdf:first, rdf:rest, rdf:nil, rdf:List`), describe structured values (`rdf:value`) or containers of values (`rdf:Seq, rdf:Bag, rdf:Alt,etc`…) without making any strong restriction/conditions on it.

RDF also introduces a reification mechanism used to make statements about other statements by turning statements into resources. In the general form the triple `<subject,predicate,object>` can be *reified* under the resource “`StatementAboutStatement`” as follow:

RDF is a universal generic language not suited to define semantics for any particular application domain. For that purpose, RDFS (RDF-Schema), the next layer in the Semantic stack on top of RDF, has to be used.

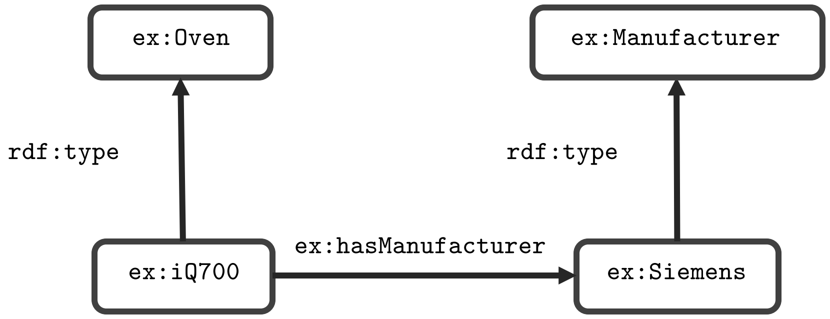


Figure 4 : Structuring RDF graph

2.2.3 RDFS

RDFS extends RDF vocabulary and introduces basic elements (class, class hierarchy, inheritance, properties and properties hierarchy), helping the knowledge description model designer to describe and classify the concepts, the entities and the relationships of a *specific application domain*. This extended vocabulary actually allows designer to add semantic constraints linked to a particular application domain. The knowledge description model of a specific domain is called an *ontology*.
By introducing the notion of class, RDFS also introduces the notion of instances (or individuals) of a class. The relationship between class and its instance(s) is done through `rdf:type`. For instance, a simple ontology is depicted in Figure 5.

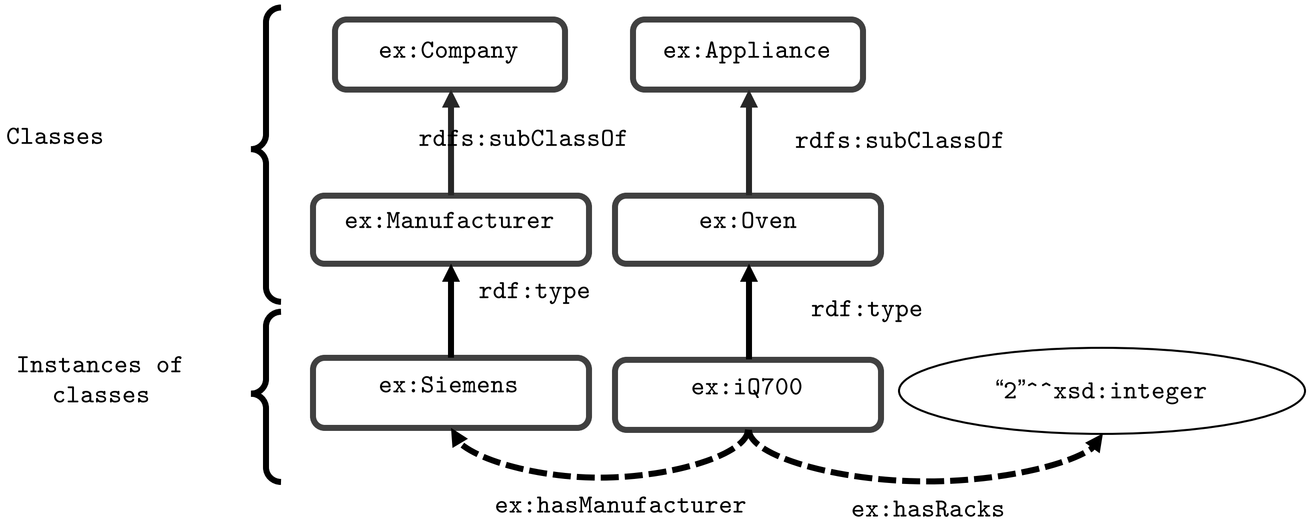


Figure 5 : A small RDFS knowledge description model

By inheritance, `ex:Siemens` and `ex:iQ700` are also respectively an `ex:Company` and an `ex:Appliance`. Additionally, RDFS introduces some mechanisms to define a signature of a relationship restraining its scope of application (note that relationships and properties as defined in Paragraph. 2.2.1 are both described using `rdf:Property`). For example, the description of the relationship `ex:hasManufacturer` can be described as follow:

The whole ontology depicted in Figure 5 is defined here after using RDFS :

2.2.4 OWL Family

On top of RDF and RDFS in the Semantic web stack, OWL (Web Ontology Language) brings much more expressivity for describing classes, properties and relationships by relying on and extending RDF fact-stating ability and RDFS class and property structuring capabilities. It allows the designer to specify a class as logical combinations of other classes (`owl:intersectionOf`, `owl:unionOf`, etc...) or to express disjointure between classes (`owl:disjointWith`). Additional class axioms can also be used to define equality between classes (`owl:equivalentClass` and `owl:sameAs`).

OWL extends `rdf:Property` semantics with `owl:datatypeProperty`, `owl:ObjectProperty`, etc... Some restrictions can be defined on the property values (`owl:allValuesFrom`, `owl:someValuesFrom`, `owl:maxCardinality`, etc...). Finally OWL adds some properties characterization mechanisms like symmetry (`owl:SymetricProperty`), transitivity (`owl:TransistiveProperty`) or inverse property (`owl:InverseOf`). Finally, ontology import is supported (`owl:import`) allowing to reuse and share ontologies.

For example [11], using RDFS a knowledge designer can:

✓ Declare classes like `Manufacturer`, `Appliance`, `Oven`,
✓ State that `Oven` is a `subclass` of `Appliance`,
✓ State that Siemens and Samsung are both instances of the class `Manufacturer`,
✓ Declare `hasManufacturer` as a property relating the class `Appliance` (its domain) and `Manufacturer` (its range),
✓ State that `hasRacks` is a property, with `Oven` as its domain and `integer` as its range,
✓ State that `iQ700` is an instance of the class `Oven`, and that its `hasRacks` has value 2.

With OWL he can additionally:

✓ State that `Manufacturer` and `Appliance` are disjoint classes;
✓ State that `Siemens` and `Samsung` are distinct individuals;
✓ Declare `manufacturerOf` as the inverse property of `hasManufacturer`;
✓ State that the class `MultipleRacks` is defined precisely as those members of the class `Oven` that have at least 2 values for the property `hasRacks`;
✓ State that `hasRacks` is a functional property (An Oven instance cannot have multiple values for rack).

These are the main capabilities brought by OWL. For a complete overview, the reader will review the OWL reference document [9] and the OWL guide [10].

The whole ontology depicted in Figure 5 (enriched with above statements) can be defined with OWL as follow:

OWL is quite complex and is derived in three sub-languages (from the less to the maximal complexity):

1) OWL Lite has a limited expressivity (class hierarchy, simple restrictions). For instance it is not possible with OWL Lite to state that an appliance has one manufacturer but can have one or multiple references names,
2) OWL DL has a high expressivity (full OWL vocabulary with some usage restrictions). For instance, with OWL DL, a class cannot be also an individual or a property,
3) OWL Full has the maximal expressivity (full OWL vocabulary without any restriction).

From a statement expressivity standpoint, OWL Lite $\sqsubset$ OWL DL $\sqsubset$ OWL Full.

OWL has been revised (OWL2 [27][28]) to overcome short-comings of OWL, such as expressivity issues, problems with its syntax, etc… OWL 2 is backward compatible with OWL and comes with two major sub-languages (OWL2 DL and OWL2 FULL) and three profiles (OWL 2 EL, OWL 2 QL and OWL 2 RL) that have restricted expressivity (but still sufficient for a large variety of applications) and thus favorable computational properties.

2.2.5 Knowledge modeling language compatibility

It is important to notice that OWL Lite and OWL-DL are not extensions of RDF language. An RDF triple is not necessarily valid in these sub-languages.

Firstly, in OWL Full, `owl:Class` is defined as equivalent to `rdfs:Class` (then all subclasses of `rdfs:Class` are also subclasses of `owl:Class`) whereas in OWL Lite and OWL DL, `owl:Class` is defined as a subclass of `rdfs:Class` (and then classes in an RDF/RDFS document that are subclasses of `rdfs:Class` cannot be a instances or subclasses of `owl:Class`).

Secondly, OLW introduces `owl:dataTypeProperty` and `owl:ObjectProperty` whereas RDFS uses `rdf:Property`. In OWL Full, `owl:ObjectProperty` is defined as *equivalent* to `rdf:Property` whereas `owl:dataTypeProperty` is defined as a *subclass* of `rdf:Property` and then also a subclass of `owl:objectProperty` (all properties in OWL defined with `owl:datatypeProperty` are also considered as owl:objectProperty). In OWL Lite and OWL DL, owl:ObjectProperty and owl:datatypeProperty are defined as disjoint to rdf:Property.

2.2.6 Summary: The Modeling Layers

We have reviewed so far the Semantic web technologies (standard vocabularies RDF/RDFS and OWL) a knowledge designer can use to formally describe the set of terms and relationships of a specific domain knowledge model (specific vocabulary, ontology). Thus, standard vocabularies is the meta model of the specific vocabulary. Knowledge designers give an abstraction model of the real world by stating a set of facts on the individuals defining a model. This model is built upon the specific vocabulary of terms and relationships which act as the meta model of the model. The Figure 6 (from [15]) depicts the complete overview of these abstraction layers.

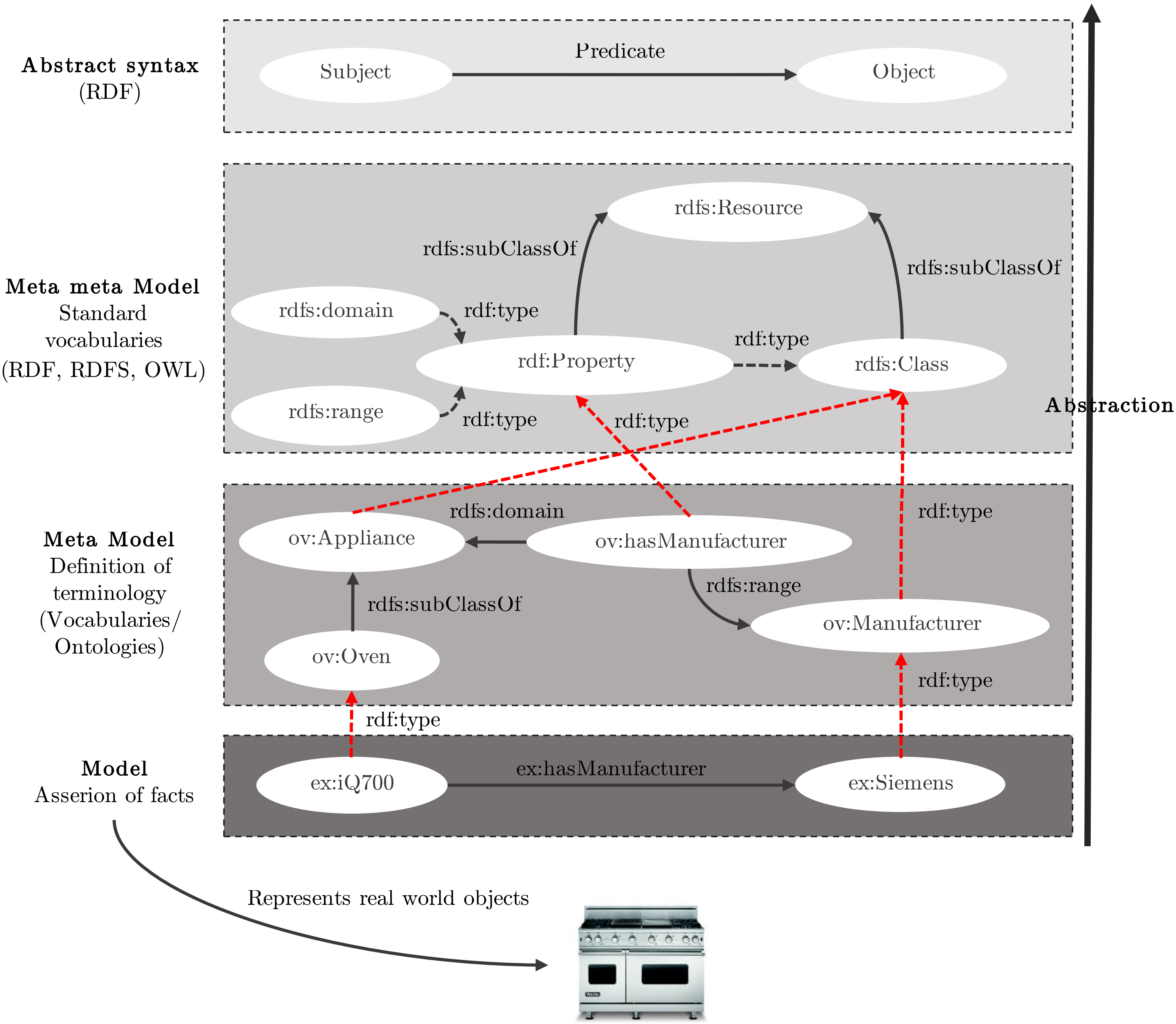


Figure 6 : Knowledge modeling abstraction layers

2.3 What knowledge in the context of ambient environments?

We have given so far a short introduction on the main semantic web technologies aimed at describing and model a knowledge and their associated modeling layers. Now let's define what type of knowledge has to be modeled in order to help ambient systems, based on this knowledge, to select *the relevant* devices and services and then better assist the users.

From an architecture standpoint, ambient systems are built upon software *services* integrated in *devices* placed in a *physical environment* (embedded in an object, on users, etc…). These devices implement *resources* interacting with objects (actuator) and/or gathering data (sensor) about themselves, the objects or the environment. Access to these resources is achieved through the services exposing their *interface* to the ambient application. From this description results three fundamental knowledge types (Figure 7):

1. The **contextual knowledge** denotes facts qualifying the device in its environment (aka context aware applications) gathered from *sensors*. For instance, its location, its acceleration, etc… This kind of knowledge can be associated to instance properties and values in the semantic web description language (the model layer).
2. The **structural knowledge** denotes facts qualifying the intrinsic characteristics of the device and/or the resources. For instance, its size, its category (TV, Car, Oven, Lamp, etc…), etc… This kind of knowledge can be associated to the meta model layer defining concepts and properties, relating either a concept to another (e.g. TV is a Device) or a concept to a value (TV has display size 15 inches).

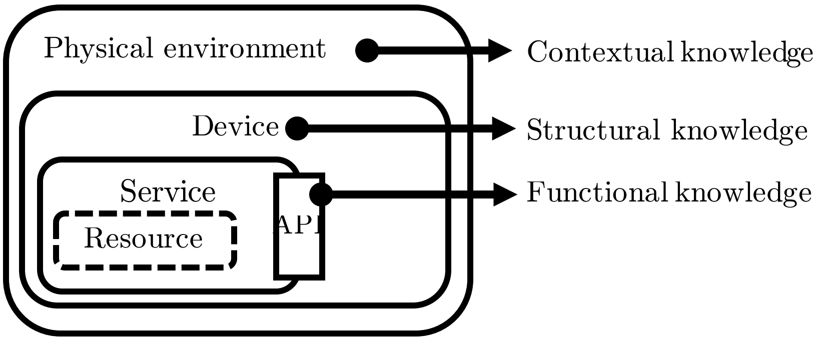


Figure 7 : Ambient environment knowledge types

3. Finally, the **functional knowledge**, that is the web services APIs, have to be described (method names, input types, output types). This is necessary: (1) to give a semantic of the purpose/usage of the services (functionality, pre-conditions, post-conditions, etc…), (2) as the web services have to work in concert, some web services output are input to some others and interoperability verification are needed (data types).

As being related to real world environments and objects, the ambient systems underlying knowledge model has to be as expressive as possible to give the knowledge designer the capacity to describe the world components and express statements about them as accurately as possible. This is key in order to minimize the abstraction level and then, increase the chance, for the ambient systems, to efficiently segregate devices and services and select the most relevant ones.

2.4 Which Meta Meta Model to Use?

At a first glance, one can say that the more a language is expressive the better it is to accurately shape a model of a knowledge domain. So, OWL family and more particularly OWL Full seems to be the best choice for describing the complex knowledge about ambient environments (Paragraph 2.3). At this point, it seems appropriate to discuss the *reasoning* capabilities of these languages (linked to their expressivity) and their impacts in term of computing complexity (and hence in term of the whole ambient system responsiveness). Indeed, the choice of the modeling language has (also) to be driven by the constraints inherent to the "embedded" characteristic of the ambient systems and then their relatively low computing resources.

Reasoning (see also Paragraph. 4.3) is the process of deriving facts that are not explicitly expressed in the ontology (inferencing). First-order logic (FOL, also called predicate logic) is the formal foundation of the

OWL language and is good at representing the knowledge (based on triples) and reason on it. Actually, from a logical standpoint, OWL classes are **unary predicates** (e.g. Device(x) standing for x is a Device), properties are **binary predicates** mapping an instance of a class (the domain of the relationship) to instances of a class or datatype (the range of the relationship), and **constraints** which could be translated to logical formulas. The issue with FOL is that the logical *entailment* computation problem is not decidable (It doesn't exist a computational process that solves the problem in a finite number of steps). Description logics (DLs) [23][24][25][26] are fragments of FOL which have been created to overcome FOL decidability issue (DLs have restrictions on logical formulas thereby ensuring decidability). Each DL brings more or less expressivity power and reasoning capabilities (computational complexity) depending logics formulas supported.

DLs are composed by basic descriptions (the atomic concepts (A,B,…), the atomic roles (R,…) and the individuals) and constructors (Table 1) from which can be built complex concepts (C,D…) and roles descriptions. The DL *syntax* is given by its signature and the constructors. The example in Figure 5 can be represented in DL as follow:

In the example above, Manufacturer, Company and Oven are concepts names, hasManufacturer is a role name and iQ700 and Siemens are individual names. ⊓(intersection) and ∀(value restriction) are constructors. Table. 1 summarizes the main DLs ($\mathcal{AL}$, $\mathcal{ALC}$, $\mathcal{S}$) and extensions ($\mathcal{I}$,$\mathcal{H}$,$\mathcal{F}$, etc…) and associated constructors along with OWL syntax.

DL	Constructors
$\mathcal{AL}$	⊤,⊥,A,¬A,C⊓D,∀R.⊤, ∀R.C
Extensions…	
$\mathcal{ALC}$	¬C , C ⊔ D, ∃R.C
$\mathcal{S}$	Roles transitivity See Paragraph. 4.3
+$\mathcal{I}$	R^{-1}
+$\mathcal{H}$	Roles hierarchy See Paragraph. 4.3
+$\mathcal{F}$	Functional roles See Paragraph. 4.3
+$\mathcal{N}$	(≥nR) (≤nR)
+$\mathcal{Q}$	(≥nR.C) (≤nR.C)
+$\mathcal{O}$	$\{a_1, \dots, a_n\}$

DL syntax	Description	OWL syntax
A	Atomic concept	URI
⊤	Top	owl:Thing
⊥	Bottom	owl:Nothing
C,D	Complex concept	owl:Class
R	Atomic role	owl:ObjectProperty owl:datatypeProperty
¬C	Negation	owl:complementOf
C ⊓ D	Intersection	owl:intersectionOf
C ⊔ D	Union	owl:unionOf
C ⊔ D =⊥	Disjointure	owl:disjointWith
∀R.⊤	All individuals of the domain of R	
∃R.C	Existential restriction	owl:someValuesFrom
∀R.C	Value restriction	owl:allValuesFrom
R^{-1}	Inverse role	owl:inverseProperty
nR	Number restriction	ow:minCardinality
nR.C	Qualified number restriction	owl:maxCardinality
$\{a_1, \dots, a_n\}$	Individuals	owl:oneOf

Table 1 : DLs family main constructors

OWL is built upon DLs (Figure 8) and, from a DLs standpoint, OWL family expressivity/computational complexity could be classified as depicted in the Table 2.

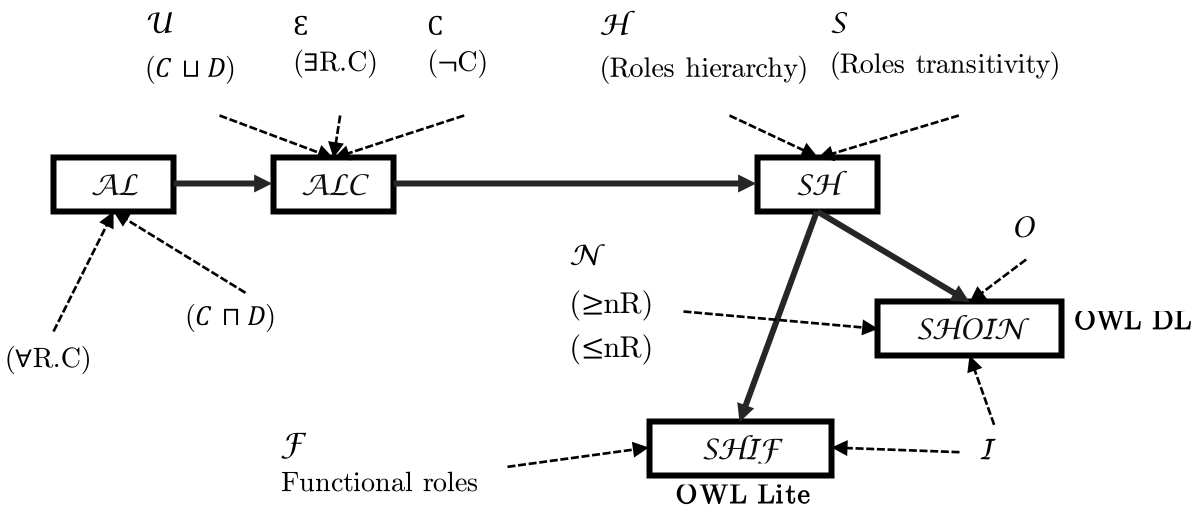


Figure 8 : OWL family DL correspondance

OWL Family	DLs Family	Worst-case reasoning* combined Complexity**	Expressivity	Comments
OWL Lite	$\mathcal{SHIF}$	EXPTIME[1]	++	
OWL DL	$\mathcal{SHOIN}$	NEXPTIME[2]	+++	
OWL Full	N/A	**Non decidable**	+++++	
OWL 2 DL	$\mathcal{SROIQ}$ [29]	N2EXPTIME[3]	++++	
OWL 2 EL	$\mathcal{EL}$++	PTIME[4]	++	For applications that need to create ontologies with very large number of classes and/or properties (*TBox*)
OWL 2 QL	*DL-Lite based*	NLOGSPACE[5]	+	For applications that want to reason on top of very large volumes of data (*ABox*)
OWL 2 RL		PTIME	-	For applications that want to describe rules in ontologies.

*Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking (See Paragraph. 4.3)
**The complexity measured with respect to both the size of the axioms, the size of the assertions

[1] EXPTIME : The set of problems resolved by deterministic Turing machine using at most exponential time
[2] NEXPTIME : The set of problems resolved by non-deterministic Turing machine using at most exponential time
[3] N2EXPTIME : The set of problems resolved by non-deterministic Turing machines using at most double exponential time.
[4] PTIME : The set of problems resolved by deterministic Turing machine using at most polynomial time
[5] NLOGSPACE : The set of problems resolved by Turing machines using at most logarithmic working memory
Note: NLOGSPACE ⊆ PTIME ⊆ EXPTIME ⊆ NEXPTIME ⊆ N2EXPTIME

Table 2 : OWL family expressivity vs computing complexity comparison

The choice of the knowledge modeling language to be used (the meta meta model) to describe the devices and the services (the meta model and the model), taking into account the high modeling language expressivity needed in the context of ambient environments and the reasoning decidability, leads to choose OWL*(2)* DL.

2.5 The Open World Assumption

In [4], authors have identified the following characteristics of the semantic web knowledge description languages:

1) **Anyone can say Anything about Any topic (AAA)**. A knowledge model designer is free to write any assertions (the model in Figure 6) envisioning a meaning whose terms (subject, predicate and object) are either defined by himself or, as often the case and promoted by the W3C, rely on known published vocabularies[6] (the meta model in Figure 6). Note that the authorities publishing these vocabularies cannot enforce the way it has to be used. Thus, it is necessary, for the designer, to get an expertise of

[6] http://lov.okfn.org/dataset/lov/

2) The domain to be modeled or rely on a domain expert familiar with the underlying semantics of the terms. While it cannot be completely avoided, this would limit the semantic heterogeneity issue that may prevent different vocabularies about the same domain to cooperate as they use different concepts and relationships for the same meaning,
3) **Open World Assumption (OWA)**: due to 1), **there might be new or complementary knowledge description that we are not aware of**. Therefore, no conclusion can be drawn from the non-existence of assertions (As opposed to the Close World Assumption (CWA) where missing assertions are considered false),
4) **Non unique naming assumption**: Importantly, as designer can locally define URIs, the same concept can be represented by different URIs across several ontologies (in that case, OWL allows to define class equivalence).

These characteristics are very important in the case of their application to describe real world devices and services. In most of the current ambient applications, devices and services knowledge description models rely on *static* and *ad-hoc* vocabularies (meta model) defining and structuring all the concepts and relationships for a *specific* domain targeting specific applications (smart homes, smart cities, building automation, healthcare, etc…).

However, while this approach is a solution for handling the semantic heterogeneity issue (the meaning of all the assertions in the devices and services knowledge description models rely on a global and common vocabulary), it limits the scope of use of the information to a single applicative domain. Extending the scope of use of the information to multiple applicative domains implies to develop a *comprehensive* vocabulary describing the world from heterogeneous vocabularies which is unlikely to happen in the SWoT context where domains to cover are countless.

There are many IoT companies and connected devices manufacturers all around the world (Figure 10), and certainly much more to come (Figure 9), each of them addressing a particular domain (Figure 1) and having their own understanding and context usages of terms that it would be worth to describe through heterogeneous vocabularies (referring to AAA).

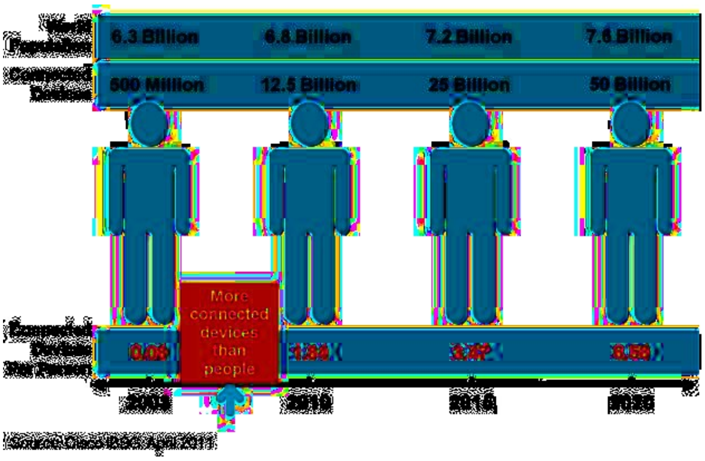


Figure 9 : IoT 2003 - 2020 projection[1]

2.6 Ontology Modeling Approaches

Some projects acknowledged the fact that multiple heterogeneous ontologies management is needed in the case of systems targeting a wide range of applicative domains. For example, in the context of ambient intelligent environments (AIEs), ATRACO project authors [12][13] envision that a comprehensive, agreed and validated ontology is unlikely to happen, and that, more realistically, device manufacturers will independently develop their own ontologies. Authors in [13], depict three possible ontology architectures to represent and manage distributed knowledge sources:

2.6.1 Global Ontology (GO) Approach

With this approach, a global domain ontology is used to formally and strictly describe *all* the concepts, the relationships and the individuals of a given domain. There is no need for metadata self-describing the devices and the services in that case. **As we have seen previously, an accepted and validated ontology describing the whole world's concepts, relationships and instances is unlikely to happen [13].** The approach and modeling levels distribution is depicted in the Figure 11.

[1] https://www.cisco.com/web/about/ac79/docs/innov/IoT˙IBSG˙0411FINAL.pdf

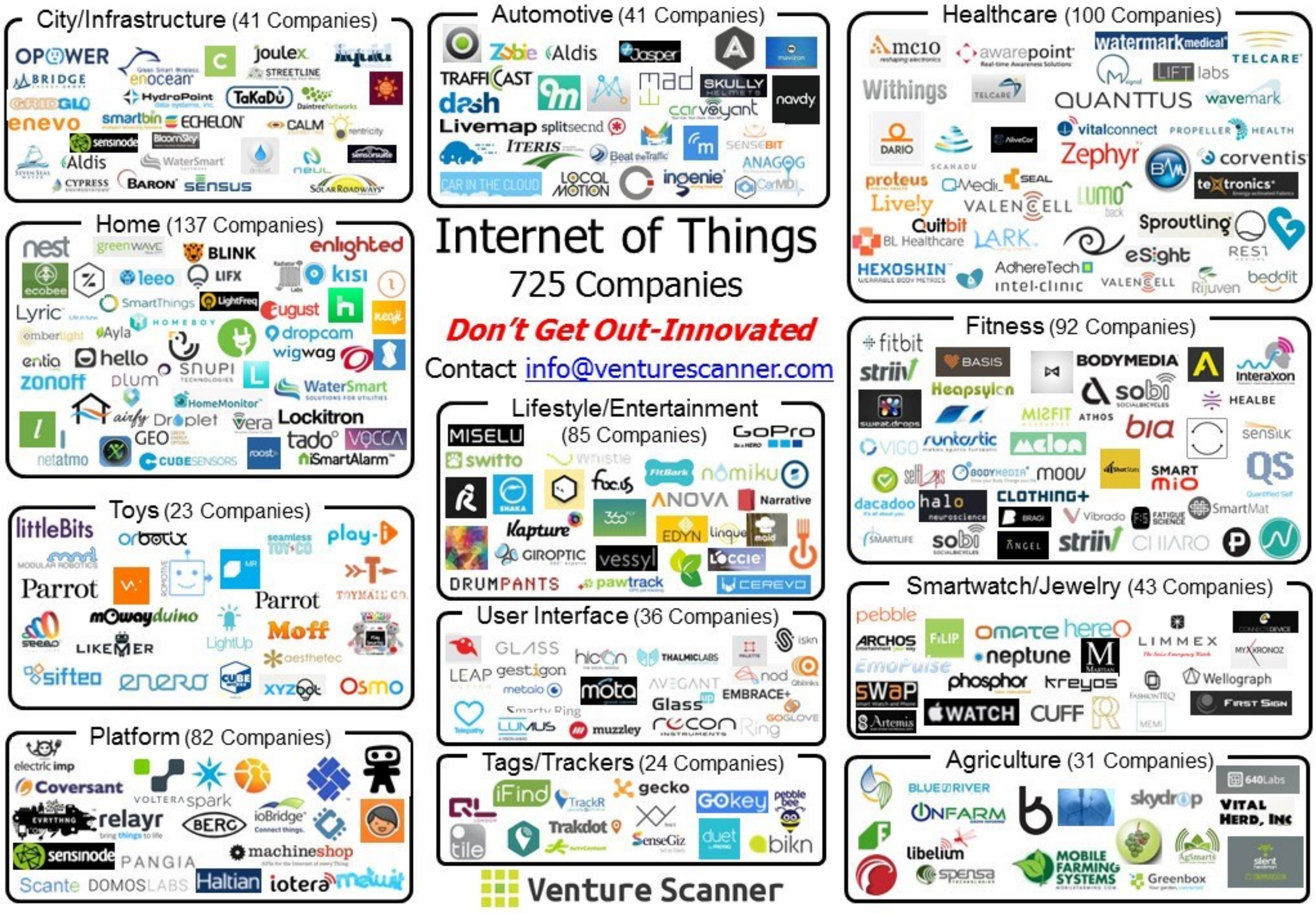

City/Infrastructure (41 Companies)
Automotive (41 Companies)
Healthcare (100 Companies)
Home (137 Companies)
Internet of Things
725 Companies
Don't Get Out-Innovated
Contact info@venturescanner.com
Fitness (92 Companies)
Lifestyle/Entertainment (85 Companies)
Toys (23 Companies)
User Interface (36 Companies)
Smartwatch/Jewelry (43 Companies)
Platform (82 Companies)
Tags/Trackers (24 Companies)
Agriculture (31 Companies)
Venture Scanner

Figure 10 : IoT companies and device manufacturers

2.6.2 Multiple Local Ontology Approach (MLO)

With this approach, each device, through embedded metadata, embeds its own domain ontology based on its own vocabulary (referring to AAA). This approach is the best fitting with our problem statement as it allows knowledge publishers to rely on their own vocabulary. The approach and modeling levels distribution is depicted in the Figure. 8.

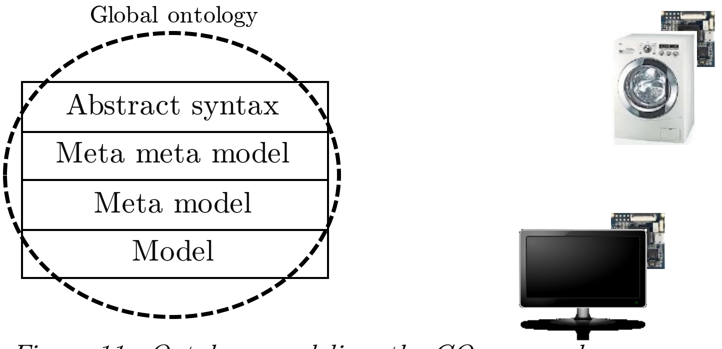


Figure 11 : Ontology modeling, the GO approach

Some projects make use of heterogeneous ontologies. For example, in the context of ambient intelligent environments (AIEs), ATRACO project previously discussed [12][13] is built around agents exchanging data between each other. This project is based on an upper ontology (Paragraph 2.6.3) but allows software agents to independently and locally describe and rely on their own ontology.

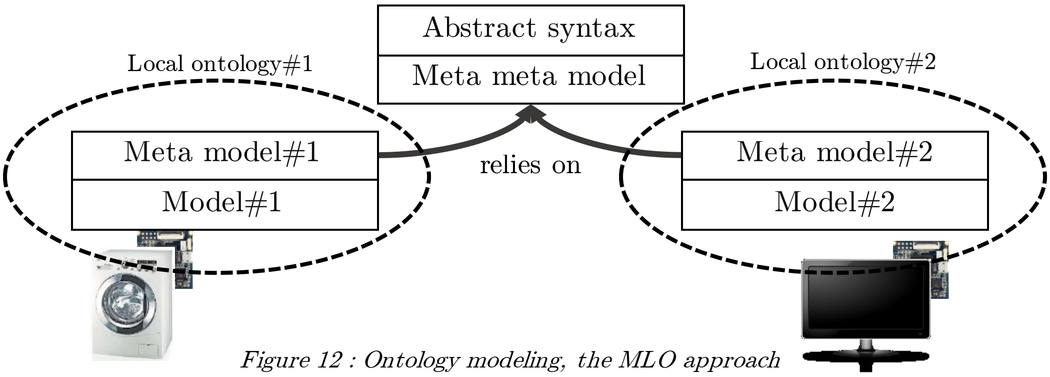


Figure 12 : Ontology modeling, the MLO approach

2.6.3 Hybrid Approach

Each device defines, through embedded metadata, his own domain knowledge (Model) built from a common vocabulary (Meta model + Meta meta model, aka Upper ontology). Like for the global ontology approach depicted in the Paragraph 2.6.1, there is currently no agreed and widely used common and global vocabulary (world comprehensive meta model) available. The approach and modeling levels distribution is depicted in the Figure 13.

For example, in [16] authors have defined layered ontologies defining a common ontology from which semantic annotations can be defined and deployed on devices. The authors highlight the need for a standardization committee and the need, for the manufacturers to develop their device ontologies based on the defined and agreed vocabulary. As it is a good solution to cope with the semantic heterogeneity issue, in SWoT context, it is unlikely that such a standardization could occur.

Most of the projects rely on ad-hoc ontologies specific to domain like smart offices [17], smart homes [18], ambient assisted living [19], sensors [20], [21] , smart cities [22], etc…

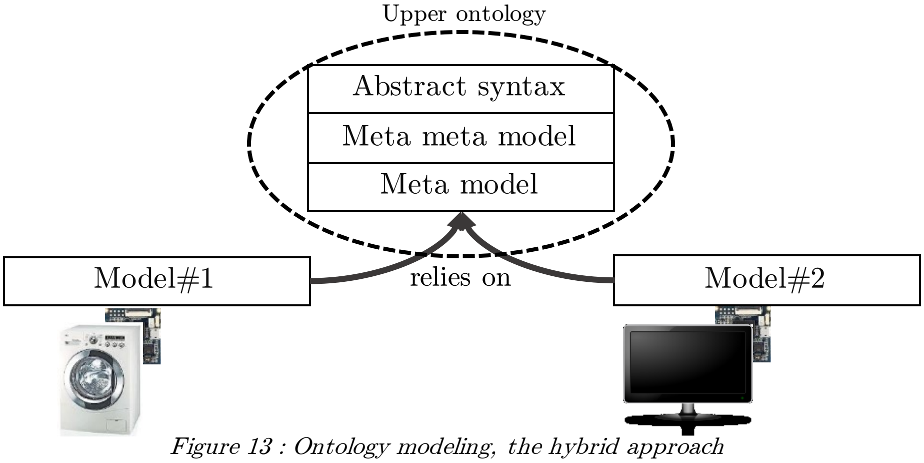


Figure 13 : Ontology modeling, the hybrid approach

2.6.4 Fragmented Hybrid (FH) Approach

This approach can be placed in between the MLO and the Hybrid approach. An upper ontology is fragmented in multiple sub-ontologies defining independently each concept of the global ontology (Paragraph. 2.6.1). It is depicted in the Figure 14. Thus, given an upper ontology U defining C concepts, $U = Sub_1 \cup Sub_2 \dots \cup Sub_n \; where \; n = C$. Since both approaches (MLO and FH) can be used together and are conceptually closed, except if it is explicitly denoted FH, we will consider (and denote) FH as MLO in the rest of the document.

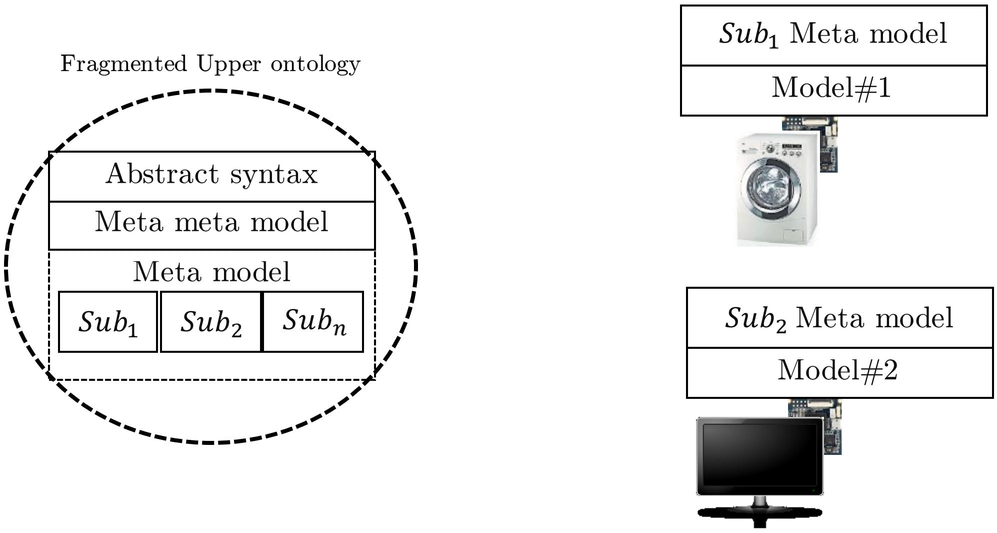


Figure 14 : Ontology modeling, the FH approach

2.7 Conclusion

We have briefly reviewed in this chapter the main web semantic knowledge modeling technologies used to formally describe knowledge and their capabilities for expressing, with more or less expressivity, facts about the real world (Paragraph 2.2). We have then defined (Paragraph 2.3), in the context of ambient systems, the knowledge types to be described: (1) the contextual knowledge, (2) the structural knowledge and (3) the functional knowledge. Such a variety of knowledge domains to cover enforces the need for a high knowledge model expressivity. While expressivity is key at reducing the knowledge abstraction level and then helps ambient systems to better segregate devices and services, it implies computational complexity that may be problematic for ambient systems with low computational resources. We concluded that OWL(2) DL is the language better fitting with the expressivity/computational complexity constraints (Paragraph. 2.4).
OWL language is built on: (1) the AAA slogan (Anyone can say Anything about Any topic) and (2) the open world assumption (there might be new or complementary knowledge description that we are not aware of). This tells us that the development of a comprehensive model of the world is not going to happen but instead, as per (1) and (2) paradigms, has to be built from the capitalization of multiple knowledge providers (Paragraph. 2.5). Unfortunately, most of the approaches, in the domain of ambient systems, relies on an *ad-hoc* knowledge model, which goes against the aforementioned paradigms. It exists a constellation of IoT providers and we envision that each of them will use their own knowledge model. After having reviewed the main ontology modeling approaches (Paragraph 2.6), we have proposed an approach based on heterogeneous knowledge representation models (MLO or MLO + FH approaches) to qualify and self-describe devices and services through semantically enriched metadata (Paragraph 2.6.2).
While this approach brings semantics heterogeneity issues the ambient systems have to cope with when integrating this heterogeneous knowledge (Chapter 3), it gives outstanding perspectives in term of services selection relevancy improvement. Indeed, as the same terms might be differently described in heterogeneous ontologies, they may acquire a more accurate meaning throughout an evolutionary process, their meaning being shaped through the assertions done by their publishers (The IoT manufacturers).

2.8 References

[1] Brachman, R., & Levesque, H. (2004). *Knowledge representation and reasoning*. Elsevier.

[2] Gyrard, A., Bonnet, C., & Boudaoud, K. (2014, March). Enrich machine-to-machine data with semantic web technologies for crossdomain applications. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on* (pp. 559-564). IEEE.

[3] Berners-Lee, T., Fischetti, M., & Foreword By-Dertouzos, M. L. (2000). Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor. HarperInformation.

[4] D. Allemang and J. Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann, 2011.

[5] G. Antoniou and F. Van Harmelen. A semantic web primer. 2nd. MIT Press, 2008.

[6] T. Segaran, C. Evans, and J. Taylor. Programming the semantic web. O'Reilly Media, 2009.

[7] Jeremy J. Carroll and Graham Klyne. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. url:http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[8] L. Masinter, T. Berners-Lee, and R.T. Fielding. Uniform resource identifier(URI): Generic syntax (2005).

[9] http://www.w3.org/TR/owl-ref/

[10] http://www.w3.org/TR/owl-guide/

[11] Horrocks, I., Patel-Schneider, P. F., & Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. Web semantics: science, services and agents on the World Wide Web, 1(1), 7-26.

[12] Goumopoulos, Christos, et al. "Atraco: Adaptive and trusted ambient ecologies." Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on. IEEE, 2008.

[13] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001, August). Ontology-based integration of information-a survey of existing approaches. In IJCAI-01 workshop: ontologies and information sharing (Vol. 2001, pp. 108-117).

[14] Kameas, A., & Seremeti, L. (2011). Ontology-based knowledge management in NGAIEs. In Next Generation Intelligent Environments (pp. 85-126). Springer New York.

[15] Holst, T. (2013). Structural analysis of unknown RDF datasets via SPARQL endpoints (Doctoral dissertation, Master thesis defense 11).

[16] Dibowski, H., & Kabitzsch, K. (2011). Ontology-based device descriptions and device repository for building automation devices. EURASIP Journal on Embedded Systems, 2011, 3.

[17] Ryu, M., Kim, J., & Yun, J. (2015). Integrated Semantics Service Platform for the Internet of Things: A Case Study of a Smart Office. Sensors, 15(1), 2137-2160.

[18] Vacher, M., Istrate, D., Portet, F., Joubert, T., Chevalier, T., Smidtas, S., ... & Méniard, S. (2011, August). The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE (pp. 5291-5294). IEEE.

[19] Jacquet, C., Mohamed, A., & Bellik, Y. (2013). An Ambient Assisted Living Framework with Automatic Self-Diagnosis. International Journal on Advances in Life Sciences, 5(1).

[20] Park, D. H., Bang, H. C., Pyo, C. S., & Kang, S. J. (2014, March). Semantic open IoT service platform technology. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 85-88). IEEE.

[21] Alirezaie, M., & Loutfi, A. (2014). Reasoning for Improved Sensor Data Interpretation in a Smart Home. arXiv preprint arXiv:1412.7961.

[22] Lécué, F., Schumann, A., & Sbodio, M. L. (2012). Applying semantic web technologies for diagnosing road traffic congestions. In The Semantic Web–ISWC 2012 (pp. 114-130). Springer Berlin Heidelberg

[23] Baader, F. (2003). The description logic handbook: theory, implementation, and applications. Cambridge university press.

[24] Horrocks, I. (2005). Owl: A description logic based ontology language. In Logic Programming (pp. 1-4). Springer Berlin Heidelberg.

[25] Krötzsch, M., Simancik, F., & Horrocks, I. (2012). A description logic primer. arXiv preprint arXiv:1201.4089.

[26] Horrocks, I., Patel-Schneider, P. F., & Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. Web semantics: science, services and agents on the World Wide Web, 1(1), 7-26.

[27] http://www.w3.org/TR/owl2-primer/

[28] Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., & Sattler, U. (2008). OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web, 6(4), 309-322.

[29] Horrocks, I., Kutz, O., & Sattler, U. (2006). The Even More Irresistible SROIQ. KR, 6, 57-67.

3. Knowledge Extraction

3.1 Introduction

Modeling an ontology from scratch is a long and costly process as it requires experts' knowledge to be abstracted in ontologies by using adequate terms, properties and relationships. Also, has we have seen in Chapter. 2, semantic web knowledge modeling languages are complex and subtle, and there might be numerous ways to model the same knowledge. So, it is a safe bet that manufacturers, to reduce development costs, will enforce the knowledge reusability through the development of interoperable knowledge fragments self-describing the devices and the services (Assuming that manufacturers will be the knowledge publishers).

It already exists a constellation of formally described knowledge freely available all over the web. However, while formally described, this knowledge cannot be used "as is" to annotate the devices since: (1) it is often drowned in an Upper ontology or spread out in a web page and (2) does not follow the semantic web best practices [8], limiting, *de facto*, the reusability of their information outside their initial scope [1].

In this chapter we first investigate how to turn the already existing formal knowledge into metadata. Providing the manufacturers with tools helping them to rapidly extract and encapsulate the knowledge into metadata is key to reduce the products design to market cycle time. In the second part of this chapter we will give a short overview of the existing web services description languages that can be used to convey the formal knowledge descriptions to the ambient systems and investigate if these languages are adapted to convey all the needed knowledge descriptions types (functional, structural and contextual).

3.2 From Knowledge to Annotations

3.2.1 Extraction From Upper Ontologies

There are numerous upper ontologies available on the web covering several domains (smart offices [2], smart homes [3], ambient assisted living [4], sensors [5],[6], smart cities [7]). Some ontologies are well-established ontologies whose terms and relationships semantics have been deeply validated. However, these ontologies being often huge and, due to the embedded systems limited resources and the high ambient systems responsiveness needed, cannot be used "as is". Therefore, an ontology designer would want to reuse, from the upper ontology, only the statements needed to qualify a given concept (a *signature*). For instance, DogOnt [11] ontology (version 3.2.11, 897 classes, 32 object properties, 46 datatype properties and 267 individuals) aims at defining the vocabulary for Intelligent Domotic Environments (IDE). It defines a set of white goods (washing machine, stove, oven, fridge, dishwasher, deep freezer, cooker, boiler, etc…) with their associated concepts hierarchy and relationships. A designer of an ontology O interested by the Fridge signature could import DogOnt ontology (hereby ensuring Fridge semantics conservation) but the resulting ontology (O ∪ DogOnt) would be much complex to work with (search, reasoning,…). The idea is for the designer of O to only extract the *fragment* (aka module, descriptive subgraph, RDF molecule, etc…) of DogOnt that just describes the Fridge concepts that have to be reused in O. The problem statement is then the following: given an upper ontology O, we want to extract the *minimal* set S of assertions ($S \sqsubseteq O$), a *module*, qualifying the concept C ($C \in O$) *without degrading* its semantics (extraction safety). This problem is well addressed in the literature [9][10] and there exists several subgraph extraction methods (***Concise bounded description***[1] (CBD) of a resource, (more generally a resource), **Minimum Self-contained Graphs** (MSGs)[27], **RDF Molecules** [28], etc…).

[1] http://www.w3.org/Submission/CBD/

Also, care has to be taken regarding the upper ontologies quality. Authors in [1] have classified up to 291 IoT based projects ontologies from which only 23 follow the semantic web best practices recommendations [8].
Module extraction is out of the scope of this document but for the purpose of experimentations (Paragraph. 4.8), we have developed a small straightforward algorithm based on (CBD based) explained hereafter.

Algorithm 1: Module extraction. Given an ontology O and a signature S, compute module O'

```
begin
   Visited := {};
   R := getResource($S$);
   sList := $O$.listStatements() where statement's subject is equal or equivalent to R;
   for all stmt ∈ sList do
       $O'$ := $O'$ + stmt;
       extract(stmt);
   end for
end;

: extract(stmt)
       if(isResource(stmt.subject) and Visited[stmt.subject] = false) {
          Visited[stmt.subject] := true;
          r := getResource(stmt.subject);
          pList := getProperties(r);
          for all p ∈ pList {
              $O'$ += p;
              extract(p);
          end for
       end if
}
```

3.2.2 Extraction From Web Pages

The World Wide Web contains a lot of *data* about products and manufacturers (from the manufacturer's products web pages or resellers). However, most of these web pages (depicting products features) are unstructured and not semantically enriched, or when enriched, does not follow the semantic web best practices [8]. One of the best practices is to publish web data as interlinked RDF graphs (Linked Data (LOD)) following some basic principles [12]:

1) Use URIs to name things,
2) Use HTTP URIs so that names can be dereferenced,
3) Return a RDF graph upon dereferencing of those URIs,
4) Use HTTP URIs in returned RDF graphs (here the interlinked vision).

(1) is about identifying the data source hence enabling the data extraction tool to check for its reliability, (4) Here the recommendation is to use the Linked Open Vocabulary (LOV[1]) as the meta model for the model describing the data in the returned RDF graphs (and not just `owl:sameAs` with possible semantic issues [15]).

There are clear advantages at using LOV along with LOD [14]: (1) Most popular vocabularies form now a core of the Semantic Web standards, (2) vocabularies rely more and more on each other through reusing, refining or extending, stating equivalences, etc… thus increasing the meaning of the terms used in those vocabularies, (3) it reduces the semantic heterogeneity (there exist up to 517 vocabularies to date manufacturers can use to describe their devices). Some tools[2] exists to transform a dataset (html, xml, csv, etc…) to LOD that can further be used to qualify data in the web pages… or directly used in the device metadata hereby extending the ontology modeling approaches presented in Paragraph. 2.6 with a new one in between the MLO approach (Paragraph. 2.6.2) and the Hybrid approach (Paragraph. 2.6.3). We call this additional approach the Multiple Local Ontology approach based on Linked Data (MLO-LD). This approach is depicted in the Figure 15.

[1] http://lov.okfn.org/dataset/lov/
[2] http://www.w3.org/wiki/ConverterToRdf

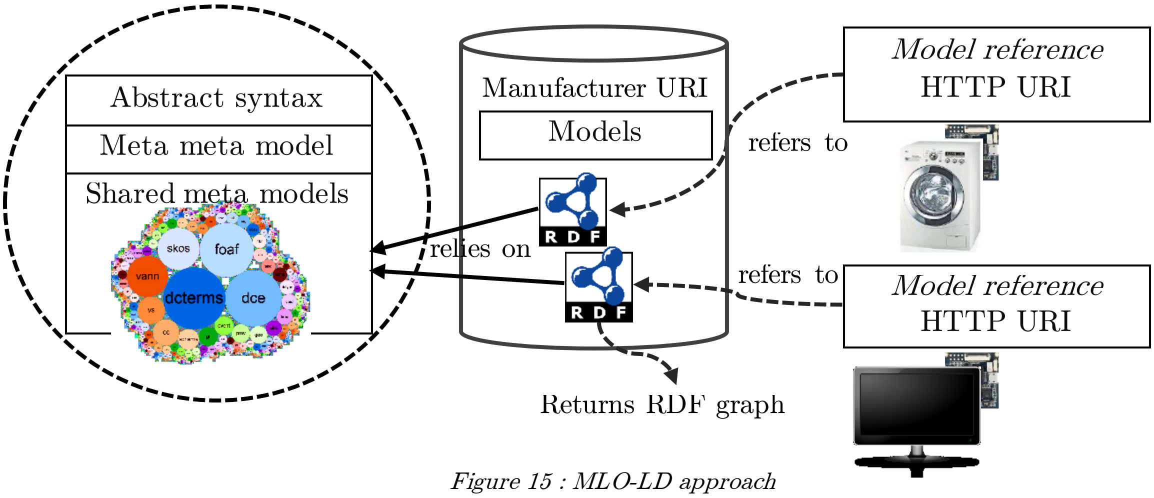


Figure 15 : MLO-LD approach

Since the knowledge is dereferenced, the main advantages of such an approach are:

1. The complete graph can be maintained by the manufacturer ensuring up to date data when retrieving it (RDF graph versioning),
2. The metadata size is reduced since just the reference to the RDF graph has to be defined.

LOVs are however not perfect. Authors in [13] noted that 3% of the LOV were not dereferenceable (one cannot retrieve the RDF graph), 25% did not maintain a creation date, 24% did not have a creator name, etc…

3.3 Web Services Standards Used to Convey the Knowledge

3.3.1 Web Services Description Languages

Ambient applications make devices and services work in concert to assist users in several distinct domains (healthcare, smart houses, etc…). This cooperation requires an underlying strong *interoperability* between devices, firstly achieved by allowing them to communicate. Among all the communication protocols, web-services (WS) based approaches (WoT, Web of Things, mainly based on WS-* (Service Oriented Architecture) or REST (Resource Oriented Architecture) approaches) is now widely accepted. W3C[1] defines a Web Service as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards".

This definition is interesting as it highlights two potential problems in the context of conveying semantically enriched metadata:

1. WSDL (Web Service Description Language) allows publishers to describe the services *functionality* through a set of operations and binding information. However, information are described *syntactically*, not *semantically*.
2. As we have seen in Paragraph. 2.3, the devices and the services knowledge to be described in the context of helping ambient systems, based on this knowledge, to select the most relevant ones, is the functional knowledge but also the structural and the contextual knowledge descriptions.

The first point (describe WSs semantically) is addressed by several services description languages (Semantic Web Services (SWS) research field), such as OWL-S [17], WSMO [18], WSDL-S [19] and SAWSDL [20]. In the rest of this paragraph, we investigate how these services description languages can cope with the knowledge to be described other than the functional one (second point).
Author in [21] gives a classification of the several services description languages for WS-* and REST based web services and denotes two approaches[1]: (1) the top-down approach where SWS are based on upper ontologies used to describe web services, (2) the bottom-up incremental approach enriching existing web

[1] http://www.w3.org/TR/ws-gloss/

services description standards (WSDL) by adding extensions to connect the syntactic definitions to their semantic annotations. Back to our vision, (1) it is unlikely that manufacturers will rely on an upper ontology to describe their products (including their functionalities), (2) we want a fine grained services description model expressivity; so it seems that the bottom-up approach is the one to adopt in the case of MLO (Paragraph. 2.6.2) or MLO-LD (Paragraph 3.2.2) approaches.

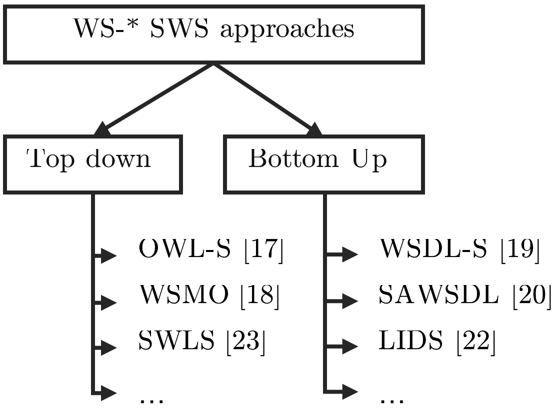


Figure 16 : WS- SWS approaches*

These description languages, primarily aims at semantically describing the functional knowledge (WS API description and usage) of the services. Non-functional knowledge representation are mainly related to the Quality of service (QoS) [25][27]. We propose then to mix-up MLO approach (or MLO-LD) with the bottom up approach (Figure 18). For information, Figure 17 depicts the MLO approach with the top down approach.

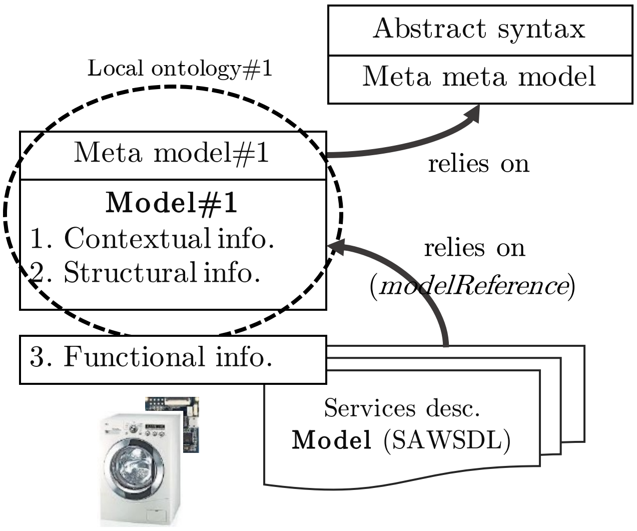


Figure 18 : MLO + Bottom up services description approach

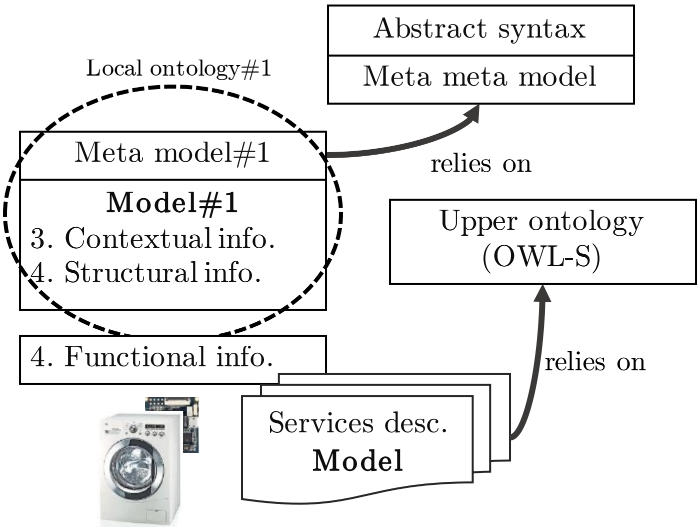


Figure 17 : MLO + Top down services description approach

[1] Unlike REST, WS-* allows to dynamically manage the devices and the services thanks to dynamic discovery (WS-Discovery) and eventing (WS-Messaging) mechanisms well suited in the context of devices embedded in real physical environments or everyday life objects (Paragraph. 4.4). This is the approach followed in this work.

3.4 Conclusion

Providing the manufacturers with tools helping them to rapidly extract and encapsulate the knowledge into metadata is key to reduce the products design to market cycle time. In this chapter we have first investigated how to turn already existing formal knowledge into metadata either from upper ontology , from which only the statements needed to qualify a given concept are extracted (a simple extraction algorithm is provided), or from web pages whose data content is published as interlinked RDF graphs (Linked Data (LD) ⟶ dereferenceable HTTP URIs) whose meta models rely on Linked Open Vocabularies (LOV). Some converter tools exists transforming any data (from html, csv, xml, etc…) to RDF triples.

From the LD approach arises a new ontology modeling approach (MLO-LD) derived from the MLO approach (Paragraph 2.6.2) which has the main advantages of (1) drastically reducing the metadata size (only a reference to an RDF sub-graph is needed in the metadata), (2) permitting manufacturers to maintain an RDF graphs repositories with possibly revisioning ensuring ambient applications to retrieve the latest RDF graph revision describing a particular device or service.

In the second part of the chapter we have given a short overview of the existing web services description languages (for SOA-based WS-*) that can be used to convey the formal knowledge descriptions to the ambient systems. Two approaches can be used depending on the situation: (1) use a top down approach where an upper ontology is used to describe the web services functionalities, (2) use a bottom up approach extending existing web services description languages (WSDL) with semantics capabilities. These description languages are mainly dedicated at defining the semantics of the services functionalities (input, output, execution flow, etc…) but not the non-functional knowledge (structural or contextual) other than the Quality of Service (QoS). We propose to mix-up the MLO approach with the bottom-up approach (assuming that manufacturers will not rely on an upper ontology to describe the functionalities of the services).

3.5 References

[1] Gyrard, A., Bonnet, C., & Boudaoud, K. (2014, March). Enrich machine-to-machine data with semantic web technologies for cross-domain applications. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 559-564). IEEE.

[2] Ryu, M., Kim, J., & Yun, J. (2015). Integrated Semantics Service Platform for the Internet of Things: A Case Study of a Smart Office. Sensors, 15(1), 2137-2160.

[3] Vacher, M., Istrate, D., Portet, F., Joubert, T., Chevalier, T., Smidtas, S., ... & Méniard, S. (2011, August). The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE (pp. 5291-5294). IEEE.

[4] Jacquet, C., Mohamed, A., & Bellik, Y. (2013). An Ambient Assisted Living Framework with Automatic Self-Diagnosis. International Journal on Advances in Life Sciences, 5(1).

[5] Park, D. H., Bang, H. C., Pyo, C. S., & Kang, S. J. (2014, March). Semantic open IoT service platform technology. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 85-88). IEEE.

[6] Alirezaie, M., & Loutfi, A. (2014). Reasoning for Improved Sensor Data Interpretation in a Smart Home. arXiv preprint arXiv:1412.7961.

[7] Lécué, F., Schumann, A., & Sbodio, M. L. (2012). Applying semantic web technologies for diagnosing road traffic congestions. In The Semantic Web–ISWC 2012 (pp. 114-130). Springer Berlin Heidelberg

[8] Semantic Web best practices: Semantic Web Guidelines for domain knowledge interoperability to build the Semantic Web of Things [Gyrard et al., OneM2M, 2014]

[9] Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2007, May). Just the right amount: extracting modules from ontologies. In Proceedings of the 16th international conference on World Wide Web (pp. 717-726). ACM.

[10] Cuenca Grau, B., Horrocks, I., Kazakov, Y., & Sattler, U. (2008). Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research, 273-318.

[11] Bonino, D., & Corno, F. (2008). Dogont-ontology modeling for intelligent domotic environments (pp. 790-803). Springer Berlin Heidelberg.

[12] Heath, T., & Bizer, C. (2011). Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology, 1(1), 1-136.

[13] Vandenbussche, P. Y., Vatant, B., & Charlet, J. (2012). Linked Open Vocabularies, un écosystème encore fragile. White paper, Mondeca.

[14] Scharffe, F., Atemezing, G., Troncy, R., Gandon, F., Villata, S., Bucher, B., ... & Vatant, B. (2012). Enabling linked data publication with the Datalift platform. In Proc. AAAI workshop on semantic cities (pp. No-pagination).

[15] Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L., & Thompson, H. S. (2010). When owl: sameas isn't the same: An analysis of identity in linked data. In The Semantic Web–ISWC 2010 (pp. 305-320). Springer Berlin Heidelberg.

[16] Tosi, D., & Morasca, S. (2015). Supporting the semi-automatic semantic annotation of web services: A systematic literature review. Information and Software Technology, 61, 16-32.

[17] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., ... & Sycara, K. (2004). OWL-S: Semantic markup for web services. W3C member submission, 22, 2007-04.

[18] De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., ... & Stollberg, M. (2006). Web service modeling ontology (wsmo). Interface, 5, 1.

[19] Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., & Verma, K. (2005). Web service semantics-wsdl-s.

[20] Kopecky, J., Vitvar, T., Bournez, C., & Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. Internet Computing, IEEE, 11(6), 60-67.

[21] Slimani, T. (2013). Semantic description of web services. arXiv preprint arXiv:1310.7367.

[22] Speiser, S., & Harth, A. (2011). Integrating linked data and services with linked data services. In The Semantic Web: Research and Applications (pp. 170-184). Springer Berlin Heidelberg.

[23] Berardi, D., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., ... & Tabet, S. (2005). SWSL: Semantic Web Services Language. Semantic Web Services Initiative (April 2005).

[24] Sheth, A. P. (2003). Semantic web process lifecycle: role of semantics in annotation, discovery, composition and orchestration.

[25] Verma, K., & Sheth, A. P. (2007). Semantically annotating a web service. IEEE Internet Computing, 11(2), 83.

[26] Klusch, M. (2008). Semantic web service description. In CASCOM: Intelligent Service Coordination in the Semantic Web (pp. 31-57). Birkhäuser Basel.

[27] Tummarello, G., Morbidoni, C., Puliti, P., & Piazza, F. (2005, May). Signing individual fragments of an RDF graph. In Special interest tracks and posters of the 14th international conference on World Wide Web (pp. 1020-1021). ACM.

[28] Ding, L., Finin, T., Peng, Y., Da Silva, P. P., & McGuinness, D. L. (2005, April). Tracking rdf graph provenance using rdf molecules. In Proc. of the 4th International Semantic Web Conference (Poster) (p. 42).

4. Knowledge integration

4.1 Introduction

With the approach proposed in Paragraph. 2.6.2, heterogeneous meta models used to describe the devices and the services are embedded in devices metadata and scattered in the environment. At some point, this knowledge has to be made available to the ambient system and *integrated* in a *knowledge base*. The knowledge integration has to cope with two major issues: (1) the physical environment and the devices dynamicity, (2) the heterogeneity of the knowledge being described through the metadata.

On the other side, such issues might be seen as a clear opportunity for the ambient system to *learn* from its interactions with the environment. Indeed, IoT manufacturers have their own understanding and context usages of terms that they use in the meta models (referring to AAA) and thus these terms, *once integrated together*, acquire a more accurate meaning over time (referring to OWA), the meaning being shaped through assertions done in the heterogeneous meta models. In ambient systems, software applications have to *make decisions* about devices and services to be used in concert to assist users. "Make a decision" is a behavior related to intelligence whose foundation is knowledge. The more accurate is the knowledge, the more *relevant* is the decision (Paragraph. 4.6).

In this chapter, we first introduce the knowledge base and its foundations (Paragraph. 4.2 & 4.3). Then, we depicts the knowledge dynamicity levels inherent to the physical nature of the environment and the devices and their impacts on the integration process (Paragraph. 4.4). Finally, we present some ontology interoperability techniques helping to cope with metal model heterogeneity issue (Paragraph. 4.5) and propose a knowledge integration model (Paragraph. 4.7) validated on two use-cases (Paragraph 4.8).

4.2 Knowledge Base

As explained in Paragraph. 2.3, description logics (DLs) underly OWL family (OWL DL is based on $\mathcal{SHOIN}$). DLs [1][2][3][4] are composed by basic descriptions (the atomic concepts (A,B,…), the atomic roles (R,…) and the individuals) and constructors (Table 3) from which can be built complex concepts (C,D…) and roles descriptions. The DL *syntax* is given by its signature and the constructors.

A DL *knowledge base* ∑ consists of a *TBox* (terminology box) and a *ABox* (assertion box) where *TBox* is a finite set (possibly empty) of general concept inclusion (GCI) axioms describing concepts (C,D,…) and roles R defining how atomic concepts (A,B,…) are related to each other's (in the form C ⊑ D, meaning D subsumes C or C ≡ D for equivalence), and *ABox* a finite set (possibly empty) of assertions describing the world state in the form C(a) (concept assertion) or R(a,b) (role assertion). Reusing the ontology presented in Paragraph. 2.4:

The TBox contains (4.1) and (4.2), and the *ABox* contains (4.3), (4.4) and (4.5).
Note that some DLs ($\mathcal{S}$,$\mathcal{H}$ or $\mathcal{F}$ based) introduce *RBox* $\mathcal{R}$ (role box), a finite set (possibly empty) of statements of the form[1]:

1. $R_1 \sqsubseteq R_2$, role inclusions,
2. Func(r), functional roles,
3. Trans(r), transitive roles.

Statements in $\mathcal{R}$ are called role axioms.

DL	**Axioms**		
	RBox	**TBox**	**ABox**

$\mathcal{AL}$		$C \sqsubseteq D$	C(a)
			r(a,b)

Extensions…			
$\mathcal{ALC}$			
$\mathcal{S}$	Trans(R)		
$\mathcal{I}$			
$\mathcal{H}$	$R_1 \sqsubseteq R_2$		
$\mathcal{F}$	Func(r)		

DL syntax	**Comments**	**OWL abstraction**
C(a)	Concept assertion	`rdf:type`
r(a,b)	Role assertion	
C ⊑ D C ≡ D	Concept inclusions	`rdfs:subClassOf` `owl:equivalentClass`
$R_1 \sqsubseteq R_2$ $R_1 \equiv R_2$	Role inclusions	`owl:subPropertyOf` `owl:equivalentProperty`
Func(r)	Func. roles (≤ 1R, >2R)	`owl:functionalProperty`
Trans(R)	Role transitivity	`owl:TransitiveProperty`

Table 3 : Knowledge base ABox, TBox and RBox axioms

TBox and *RBox* define the *general* knowledge of a given domain (established at design time by the ontology designer) and ABox, the knowledge in a *specific* situation (established at run time). Thus, one can consider *TBox* and *RBox* as a meta model and *ABox* the model (Paragraph. 2.2.6).

[1] http://www.cs.man.ac.uk/˜ezolin/dl/

4.3 Reasoning

4.3.1 Knowledge Base Interpretation

Once a DL signature S is fixed, its model semantics is given by an interpretation I. An interpretation I for S is given by $I = \langle \Delta^I, \cdot^I \rangle$ where Δ^I is the interpretation domain (set of individuals) and $\cdot^I$ an interpretation function which maps each atomic concept A to a set $A^I \in \Delta^I$, and each atomic role R to a binary relation $R^I \in \Delta^I * \Delta^I$. Each DL provides a set of constructors extending the interpretation function to give more or less semantics to atomic concepts and atomic roles (See Table 4 for the $\mathcal{ALC}$ semantics).

Constructor	**$\mathcal{ALC}$ syntax**	**Semantics**
Top	$\top$	Δ^I
Bottom	$\bot$	$\emptyset$
Negation on atomic concepts	$\neg$A	$(\neg \mathrm{A})^I = \Delta^I \setminus A^I$
Negation on complex concepts	$\neg$C	$(\neg \mathrm{C})^I = \Delta^I \setminus C^I$
Conjunction	C $\sqcap$ D	$(C \sqcap D)^I = C^I \cap D^I$
Disjointure	$C \sqcup D$	$(C \sqcup D)^I = C^I \cup D^I$
Existential restriction	$\exists$R.C	$(\exists \mathrm{R.C})^I = \{a \in \Delta^I \mid \exists y. \langle a, b \rangle \in R^I \wedge b \in C^I\}$
Value restriction	$\forall$R.C	$(\forall \mathrm{R.C})^I = \{a \in \Delta^I \mid \forall b. \langle a, b \rangle \in R^I \rightarrow b \in C^I\}$
Concept inclusion	$C \sqsubseteq D$	$(C \sqcup D)^I = C^I \subseteq D^I$

Table 4 : $\mathcal{ALC}$ constructors

An interpretation I is said to be a model of an axiom φ, or I *models* φ (written $I \vDash \varphi$), if the interpretation of φ in I is not empty. For example, let's consider ∀hasManufacturer.Manufacturer in Figure 5. $I = \langle \Delta^I, \cdot^I \rangle$ is a model of ∀hasManufacturer.Manufacturer where:

$\Delta^I = \{Oven, Siemens, Appliance, Manufacturer, iQ700, Company\}$

And the interpretation function $\cdot^I$ is defined by:

$Manufacturer^I = \{Siemens\} \neq \emptyset$
$hasManufacturer^I = \{\langle iQ700, Siemens \rangle\} \neq \emptyset$

Now, considering a knowledge base $\sum \langle T, A \rangle$ and an interpretation I, then $I \vDash \sum$ (I *satisfies* $\sum$) if and only if for all $\varphi \in T \cup A$, $I \vDash \varphi$. $\sum$ is satisfiable ($\sum \nvDash \bot$) if it exists such an interpretation I that satisfies $\sum$.

4.3.2 Reasoning Tasks

Reasoning is the process of discovering implicit knowledge *entailed* by the knowledge base $\sum = \langle T, A \rangle$. Several entailment tasks are then possible on TBox and ABox (along with RBox for $\mathcal{S}$,$\mathcal{H}$ or $\mathcal{F}$ DLs based):

a) TBox Reasoning

1. **Subsumption** ($\sum \vDash (C \sqsubseteq D)$):
   D subsumes C for a terminology T if $C^I \subseteq D^I$ for all interpretation I of T;
2. **Concept satisfiability** ($\sum \nvDash$ C $\equiv \bot$):
   A concept C of a terminology T is satisfiable if it exists an interpretation I of T such that $C^I \neq \emptyset$;
3. **Concept Equivalence**($\sum \vDash (C \equiv D)$):
   C is equivalent to D for a terminology T if $C^I \equiv D^I$ for all interpretation I of T;

4. **Disjointure** ($\sum \vDash (C \sqcup D = \bot)$):

   C and D are disjoint in a terminology T if $C^I \cap D^I = \emptyset$ for all interpretation I of T;

b) ABox Reasoning

1. **Instance checking** ($\sum \vDash C(a)$):

   For all interpretations I of $\sum$, check that C(a) holds.
2. **Role checking** (($\sum \vDash R(a,b)$):

   For all interpretations $I \vDash \sum$ check if $\left(a^i,b^i\right) \in R^I$;
3. **Knowledge base satisfiability** ($\sum \nvDash \bot$):

   Check if it exists $I \vDash \sum$.

Reasoning is used to achieve complex tasks like searching for instances of a concept, or retrieve the concept of an instance with subsumption task. Also, it is important in discovering possible issues in the knowledge base (chapter 5):

1. **Unsatisfiable concepts** ($\sum \vDash$ C $\equiv \bot$). In other words, a concept C is unsatisfiable in O if for each interpretation I of O, $C^I = \emptyset$;
2. **Inconsistency.** An ontology O is inconsistent if it has no interpretation;
3. **Incoherent knowledge base**. A knowledge base is said incoherent if it contains at least one unsatisfiable concept.

4.3.3 ∑ Saturation & Inference Rules

∑ saturation is the process of enriching *ABox* and *TBox* information using reasoning tasks and inference[1] rules on the knowledge base content. Back to the previous example (Figure 5):

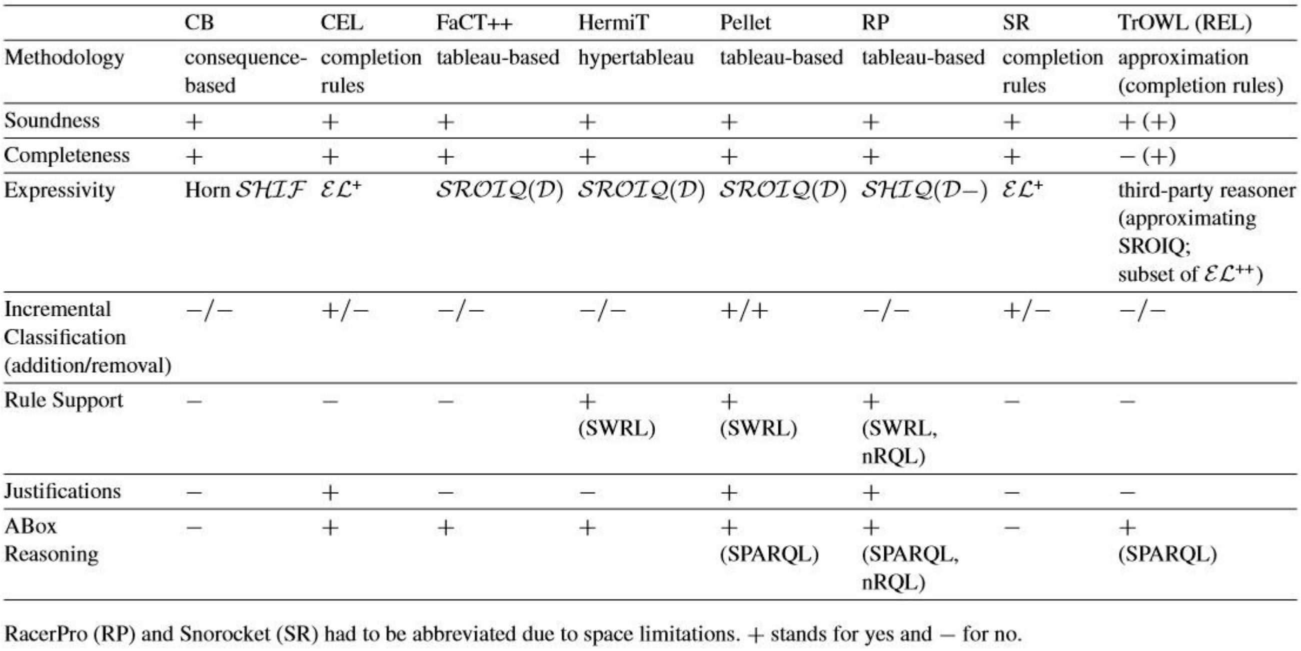
The saturated ABox is:

ABox = $\{\langle iQ700{:}\,Oven \sqcap Appliance\rangle, \langle Siemens{:}\,Manufacturer \sqcap Company\rangle, \ldots\}$

Several reasoning engines are available and can be used to compute inferences in a knowledge base. Each of them handles more or less inferencing capabilities depending on the DL expressivity supported (Table 5 from [10]).

As depicted in Table 5, some reasoners support SWRL (Semantic Web Rule Language)[5]. It extends the set of OWL axioms to enable rules to be *combined* with an OWL knowledge base. Syntax of the rule language is close to RuleML. Axioms may consist of RDF, OWL and rule axioms. The language provides pre-defined sets of built-in functions (e.g. string functions and mathematical functions, etc…). SWRL support allows knowledge designers to embed *custom* inference rules directly in the ontology (inference rules specific to a domain). For instance, considering an ontology defining a role axiom `hasPowerConsumption(a,12w)`, an SWRL rule may infer the new relation for a: `hasPowerEfficiency(a,A+)` (See 4.8.2).

[1] http://www.w3.org/standards/semanticweb/inference



	CB	CEL	FaCT++	HermiT	Pellet	RP	SR	TrOWL (REL)
Methodology	consequence-based	completion rules	tableau-based	hypertableau	tableau-based	tableau-based	completion rules	approximation (completion rules)
Soundness	+	+	+	+	+	+	+	+ (+)
Completeness	+	+	+	+	+	+	+	− (+)
Expressivity	Horn $\mathcal{SHIF}$	$\mathcal{EL}^+$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SROIQ}(\mathcal{D})$	$\mathcal{SHIQ}(\mathcal{D}-)$	$\mathcal{EL}^+$	third-party reasoner (approximating SROIQ; subset of $\mathcal{EL}^{++}$)
Incremental Classification (addition/removal)	−/−	+/−	−/−	−/−	+/+	−/−	+/−	−/−
Rule Support	−	−	−	+ (SWRL)	+ (SWRL)	+ (SWRL, nRQL)	−	−
Justifications	−	+	−	−	+	+	−	−
ABox Reasoning	−	+	+	+	+ (SPARQL)	+ (SPARQL, nRQL)	−	+ (SPARQL)

RacerPro (RP) and Snorocket (SR) had to be abbreviated due to space limitations. + stands for yes and − for no.

Table 5 : State of the art reasoners comparison [10]

4.3.4 Reasoning Over Linked Vocabulary

The MLO-LD approach (Paragraph. 3.2.2) is based on HTTP URIs permitting to retrieve RDF graph by dereferencing. From an integration standpoint, two approaches are possible:

1. Keep the HTTP URI in the knowledge base without retrieving the associated RDF graph (e.g. the concept 'TV' is defined by the HTTP URI but the RDF graph semantically defining the concept is not retrieved and integrated in the knowledge base. This approach might be used to keep the knowledge base size as low as possible or when Linked Open Vocabularies (LOV) are used as meta model hence improving the alignment process (Paragraph. 4.5.2),
2. Retrieve the RDF graph before integrating it in the knowledge base.

While the first approach is good at maintaining the knowledge base size as low as possible, it limits the reasoning capabilities and potential miss-generation of interesting inferences.

4.4 Ambient Systems Dynamicity and Impacts on the Knowledge Integration

As seen in Paragraph. 4.2, a DL *knowledge base* ∑ consists of a *TBox* axioms (terminology box), a finite set of general concept inclusion axioms (GCI) describing atomic concepts and roles defining how atomic concepts are related to each other's, and *ABox* axioms *(*assertion box)*,* a finite set of concept assertions (instances) or role assertions (properties). The knowledge base is located at the heart of the ambient systems (either fixed (Figure 20) or mobile (Figure 19)) while the knowledge it contains (*ABox, TBox*) is brought by the devices metadata in the environment, theater of physical phenomena (space, time, temperature, quality of service, etc…) implying some dynamicity at the GCI and the concept and role assertions levels. For example, in Figure 20, device states in the apartment may change, new devices may appear or disappear over time, etc… In Figure 19, new devices may be discovered according to the user (on which is attached the ambient system) movements in the city.

Figure 20 : Fixed ambient system

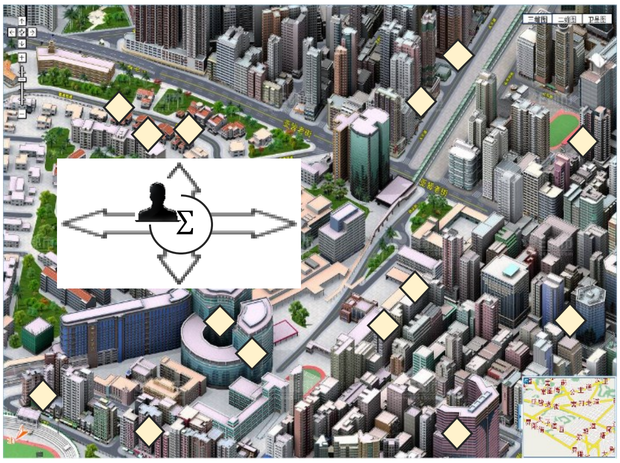


Figure 19 : Mobile ambient system

4.4.1 Role Assertion Level Dynamicity

Devices placed in the environment, worn by users or embedded in everyday life objects publish values gathered from sensors and representing the users, the environment or the objects physical states (temperature, location, battery level, etc…). For instance, in Figure 21, the device metadata brings the oven's temperature value through a role assertion (`has_temp(Siemens,180)`). The oven's temperature property in the knowledge base has to be updated according to the oven current temperature. It allows queries (See chapter 6) such as:

"*What is the current temperature of the oven?*"

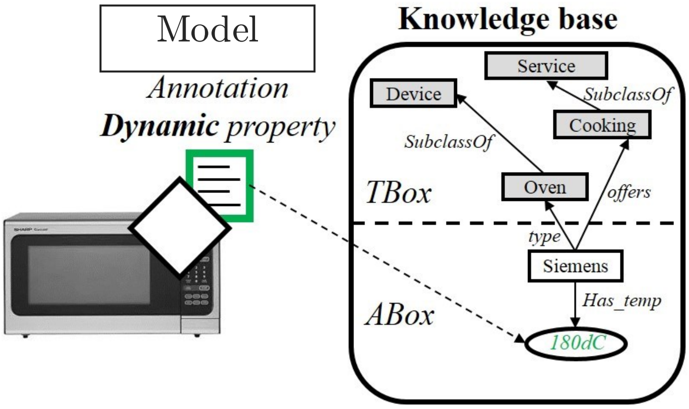


Figure 21 : The role assertion level dynamicity

4.4.2 Concept Assertion Level Dynamicity

In ambient environments, devices in the environment are not known *a priori* and unpredictably appear or disappear (Figure 22). The knowledge base must be kept up to date with the instances of the devices as they appear or disappear in the environment (For that purpose, a device discovery mechanism is necessary [11][12][13][14]). At each instant, the knowledge base content is a snapshot of the instances of the devices available in the environment (for instance `Oven(Siemens)`) permitting queries like:

"*What are currently the domestic appliances present in the kitchen?*"

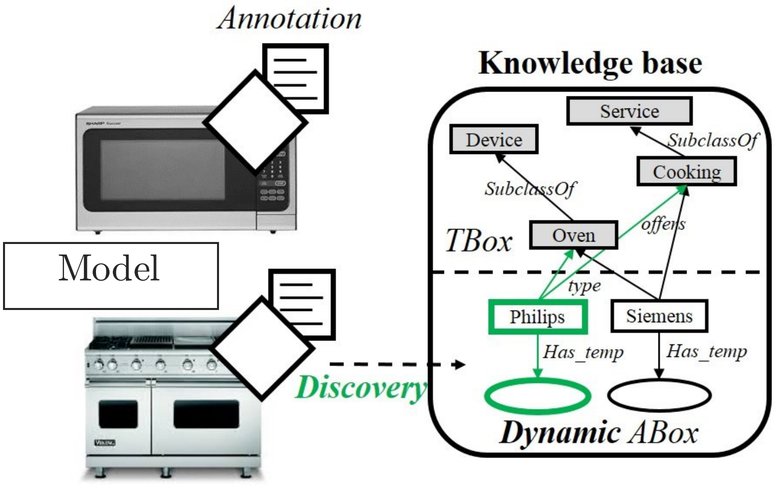


Figure 22 : The concept assertion level dynamicity

4.4.3 GCI Level Dynamicity

Based on the approach proposed in the Paragraph. 2.6.2, each device brings its meta model (Figure 23) and do not rely on an upper meta model common for all the devices.

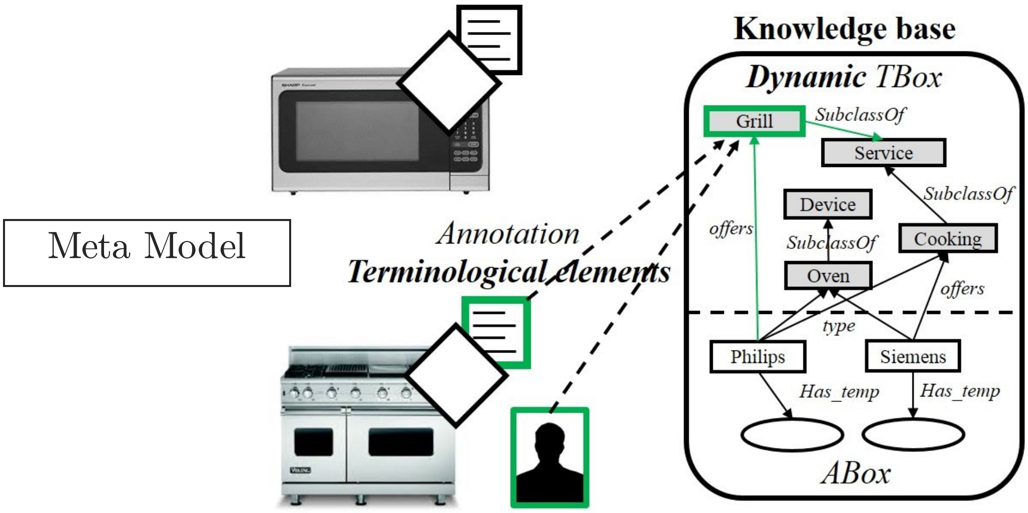


Figure 23 : The terminological knowledge

Therefore, each device is subject to *enrich* the knowledge base with new knowledge throughout the life of the system (Note that user may also enrich the knowledge base [13]). For instance, an initial query like:

"What are the domestic appliances available allowing to cook?"

corresponding to the Figure 22 would return two devices (both ovens being linked to the concept "Cooking"). If one of the device adds the new concept "Grill" (Figure 23), the initial query can be refined with:

"What are the domestic appliances available to grill?"

returning only one result (Philips oven). Note that along with additional GCI, domain specific inference rules (as discussed in Paragraph. 4.3) can also be part of the meta model helping to refine the knowledge by inferring new roles.

4.5 Heterogeneous GCI Integration Issues

As discussed in Paragraph. 4.4.3, the GCI level dynamicity enriches the knowledge base with new concepts and roles descriptions. Therefore, allowing the device knowledge description model to rely on its own meta model (heterogeneous meta model), leads the knowledge base to cope with:

1) Semantic heterogeneity issues (AAA slogan, Non unique naming assumption);
2) Unsatisfiability;
3) Incoherency;
4) Inconsistency.

An aggravating factor is the necessity, for the knowledge base, to be managed *autonomously* (possibly without (or very limited) human interaction). Some ontology interoperability techniques are available to reduce the aforementioned issues (and then allowing ontology reusability) and are explained here after.

4.5.1 Ontology Merging & Integration

Ontology *merging* is the process of generating a single, coherent ontology from two or more existing and different ontologies related to the same (or very similar) domain [15]. Ontology *integration* is the process of generating a single ontology in one subject from two or more existing and different ontologies in different domains [15] (Figure 24). Merge/Integration is an automatic process.

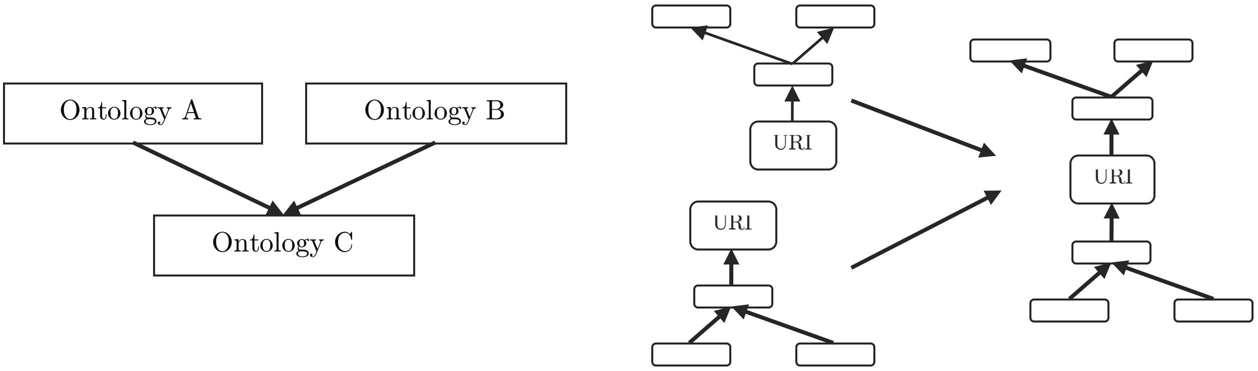


Figure 24 : Ontology Merging & Integration

4.5.2 Ontology Alignment and Mapping

Ontology *alignment* process takes two or more input ontologies and produces a set of correspondences between concepts that match *semantically* with each other (Ontology matching is the process of discovering **similarities** (∈ ℝ) between two ontologies). These matches are also called mappings [16]. Ontology alignment is made if the sources become consistent with each other (describe how the concepts in the different ontologies are logically related) but are kept unchanged (without changing the meaning in the original ontologies) [15] (Figure 25).

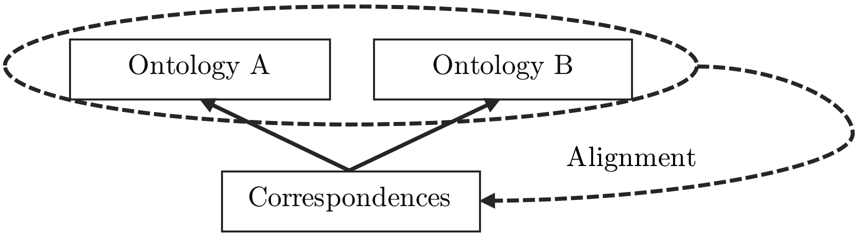


Figure 25 : Ontology alignment and mapping

Alignments can be of various cardinalities: 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) or n:m (many to-many). A correspondence is a 4-uple: (id, e1, e2, r) where id is the correspondence identifier, e1 and e2 are entities (e.g. classes and properties of the first and the second ontology), and r is a relation (equivalence (≡), more general (⊒), disjointure(⊥), etc…) holding between e1 and e2.

Correspondences have some associated metadata, a frequently used one is a confidence level (called alignment threshold, in the [0,1] range) in the correspondence results [18]. Alignment engines usually computes an aggregation of the syntactic and the semantic similarities between e1 and e2 given by the relation r [19][20].

Alignment process requires a user feedback who have to acknowledge the correspondences based on their associated confidence level before integrating it in the knowledge base. This process can be made automatic if one accept to integrate all correspondences whose confidence level is greater than a given threshold value.

Ontology matchers are not perfect and along with benchmarking [18], OAEI (Ontology Alignment Evaluation Initiative) aims at monitoring existing ontology matchers results over years [21].

The MLO-LD approach (Paragraph. 3.2.2) is a good tradeoff at limiting the semantic heterogeneity issue and improving the alignment process as the meta model relies on a set of shared vocabularies rather than completely heterogeneous ones.

4.5.3 All Together

The Figure 26 depicts the full GCI integration process in the knowledge base based on aforementioned techniques.

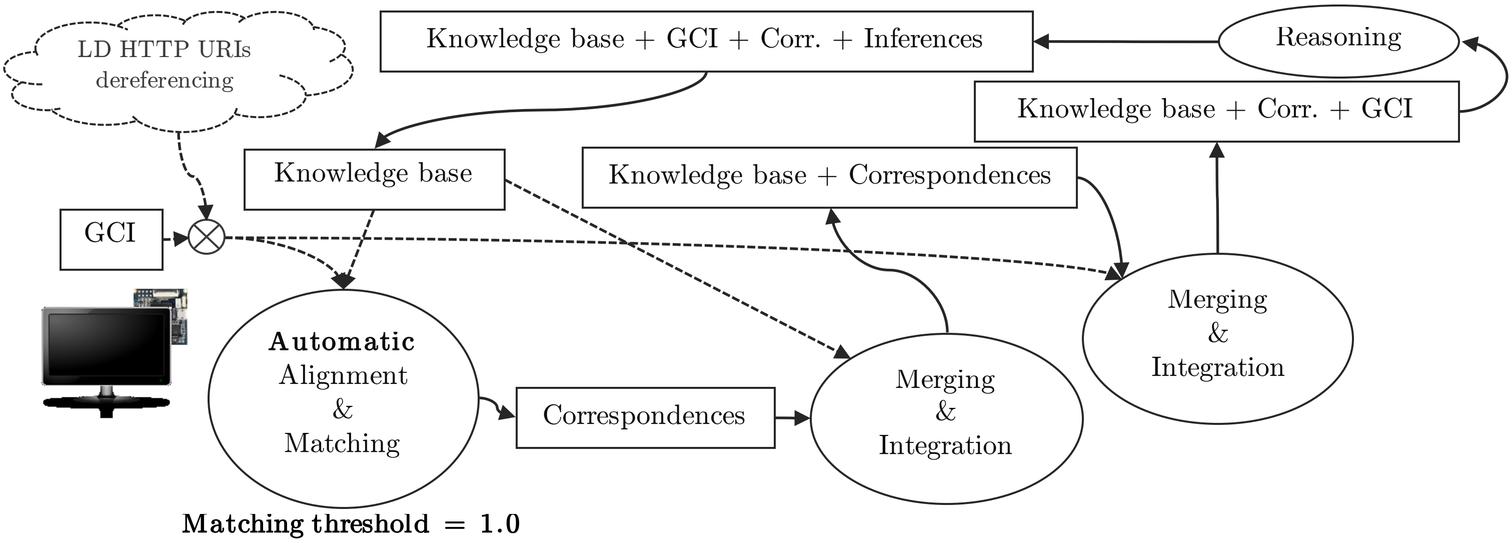


Figure 26 : GCI Integration process over time

4.6 The Semantic Heterogeneity is Good!

As explained in Paragraph. 2.2, Semantic modeling is the process of interpreting the world (or specific domains of interest) by asserting statements about the world, in other words, enouncing constraints on the possible world states. But it is unlikely that one can produce a comprehensive set of statements fully describing the world. Allowing devices metadata to rely on heterogeneous meta models (AAA) to describe themselves and their services leads the knowledge base *GCI* to increase and say more things about the world. The bigger is the knowledge, the lower is the set of interpretations holding 'true'. IoT manufacturers have their own understanding and context usages of terms that it would be worth to describe in vocabularies (referring to AAA) and thus get terms acquire a more accurate meaning through an evolutionary process (referring to OWA), the meaning being shaped through assertions done by their publishers. While the knowledge modeling language expressivity is key at allowing accurate and subtle world states to be described, allowing the knowledge evolution over time is key at improving the relevancy of the selected services (The bigger is the knowledge, the lower is the set of interpretations holding 'true').

The concept and role assertions dynamicity level is well addressed in the SWoT community (context awareness [31], sensor networks [32], etc…). The GCI dynamicity level, and hereby the knowledge base enrichment over time, on the other side, is not well considered. Some projects make use of heterogeneous ontologies. For example, in the context of ambient intelligent environments (AIEs), ATRACO project [22] is built around agents exchanging data between each other. This project is still based on an upper ontology but allows software agents to independently and locally describe and rely on their own ontology. While an ontology alignment engine is developed to cope with the semantic heterogeneity issue at run time, it still offers no perspective for the upper ontology to capitalize the contribution of agents' local ontologies over time. In [23] authors expose some challenges relative to SWoT domain. One of the identified challenges, is the ability, for the smart products, to be able to learn new emergent knowledge. But authors have been focused on emergent knowledge brought from user's interactions and feedbacks (user's preference learning) or from wiki pages, not from devices knowledge contributions. In [24] authors address the problem of gathering knowledge in order to improve user's interactions with smart products. They propose to use semantic annotations to enrich smart products workflows aimed at defining tasks and participants in several contexts. Authors highlight the problem of the domain ontologies shipped with smart products that have to be enriched over time with the knowledge about user's environment and interests. They consider possible changes at the ontology level (ontology extension) and the instance level (ontology population). The instance level described here corresponds to the knowledge base level. While the authors motivate the need of such knowledge evolution, no automatic mechanism is proposed for the enrichment other than manual.

4.7 Knowledge integration model

The knowledge integration model is depicted in Figure 27. As soon as a new device is discovered in the environment, the GCI defined in its metadata are integrated in the knowledge base). Concept assertions and role assertions (*ABox*) are integrated in the knowledge base as well but are maintained separately. When the device is not available anymore, only the *ABox* are removed from the knowledge base. The *ABox* integration

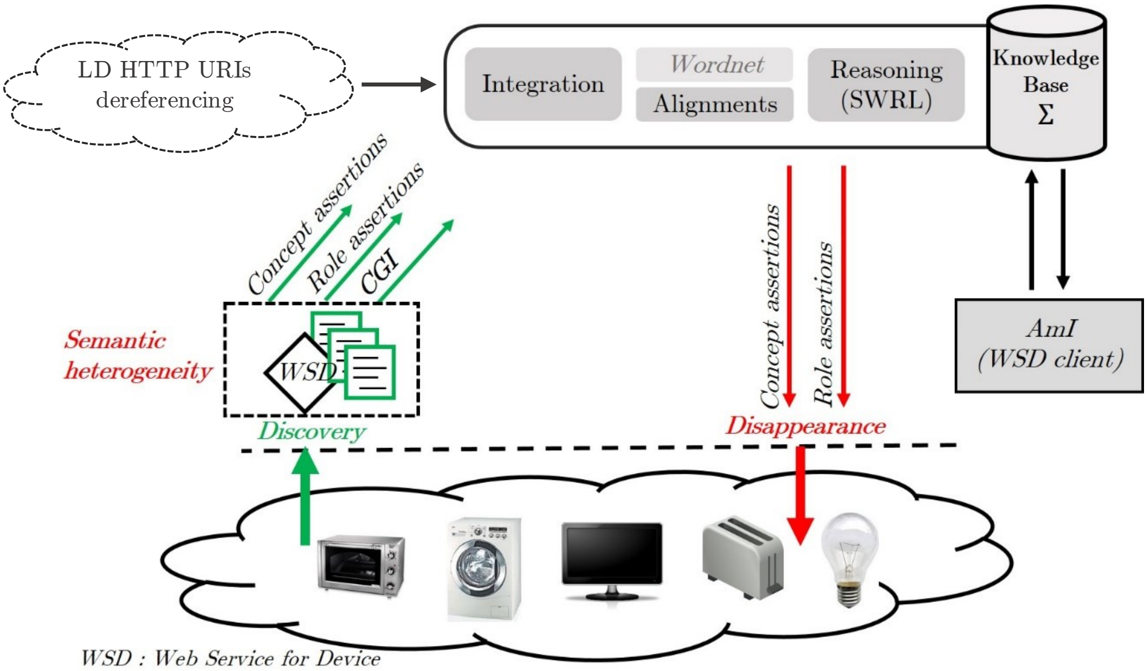


Figure 27 : Knowledge integration model

methodology is different compared with the GCI integration methodology and is depicted in Figure 28. While inferences are not an issue for GCI (as it has to be made persistent), it is not the case for *ABox*. Actually, it is easy to keep track of the integrated *ABox* but much more difficult to keep track (and then remove it from the knowledge base) the inferences computed from *ABox* addition. In order to minimize *ABox* inferences persistence issue, we first compute inferences on the *ABox* (local reasoning) before integrating *ABox* in the knowledge base. Doing so, we can keep track of the computed local inferences. Then, alignment is performed and the mapping merged with the knowledge base. Here, the mapping is isolated as well and can be removed upon device disappearance. The set ⟨*ABox*, local reasoning⟩ is then merged in the knowledge base and a global reasoning process is executed. Inferences made at this stage are difficult to track and it would be a time consuming task (keep track of the differences between the knowledge base content before and after the global reasoning).

From an implementation standpoint, either:

1) The global reasoning is enabled for ABox integration with the risk of overloading the knowledge base over time as global inferences are not removed and are then persistent,
2) The global reasoning is enabled but inferences are recorded, so some extra computing time would be required with possible impacts on the ambient system responsiveness,
3) The global reasoning is disabled with the risk of losing interesting information.

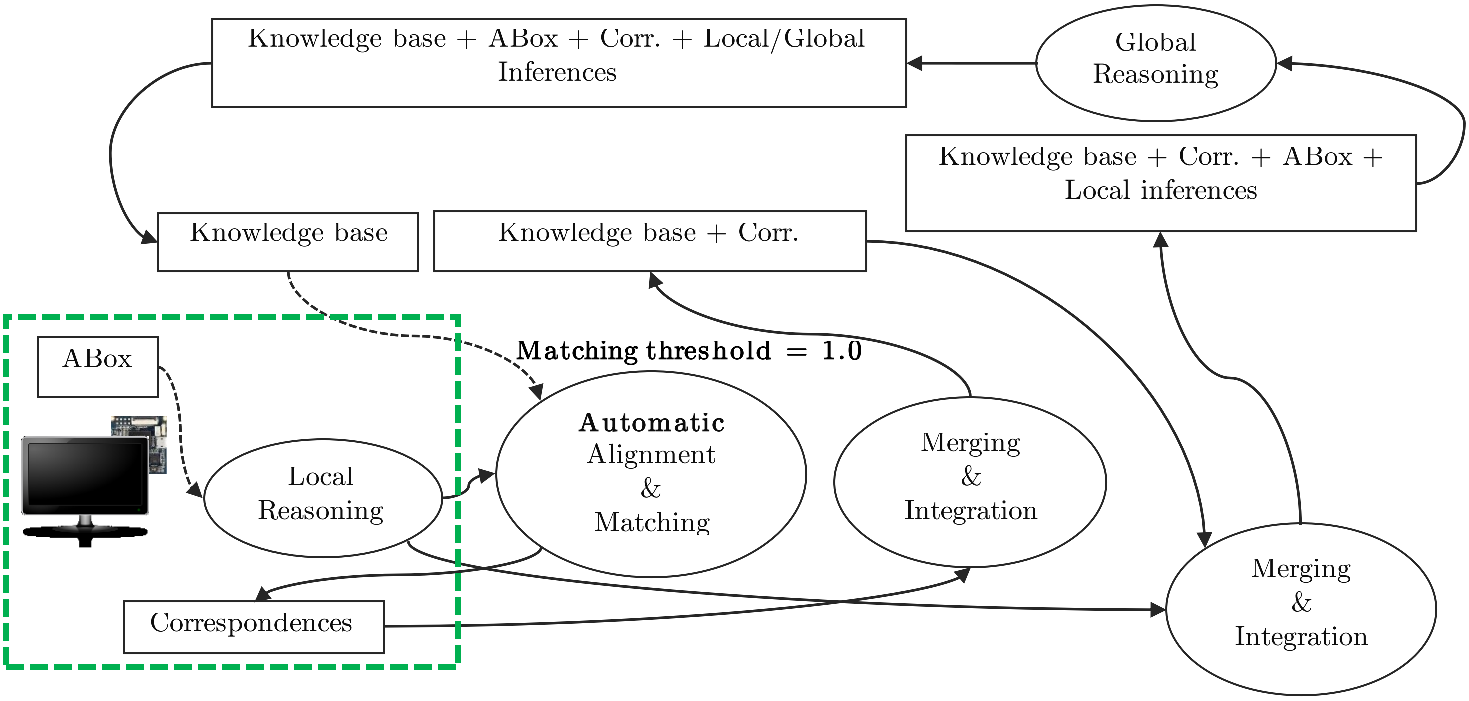


Figure 28 : ABox integration process over time

The GCI dynamicity level allows the knowledge base to be enriched throughout the life of the system. Based on this, and in the SWoT context, we classify here after (Figure 30) the ontology management approaches described in Paragraph. 2.6 according to two criteria: (1) their capacity at managing the semantic heterogeneity, (2) their faculty at permitting the knowledge enrichment over time.

1. **FH**

   With this approach, devices semantic annotations bring fragments of a meta model (the upper ontology). The GCI grows as devices are discovered over time. The GCI enrichment is bounded to the content of the meta model the fragments are extracted from, limiting *de facto* the GCI enrichment perspectives. On the other side, there is no semantic heterogeneity issue.

2. **MLO**

   With this approach, each device locally defines and embeds its own meta model. In the context of SWoT, although good at supporting GCI enrichment, the lack of a common vocabulary leads the necessity of implementing ontologies alignment mechanisms in order to smooth the semantic heterogeneity. The lack of a common vocabulary may lead to degrade the reasoning process.

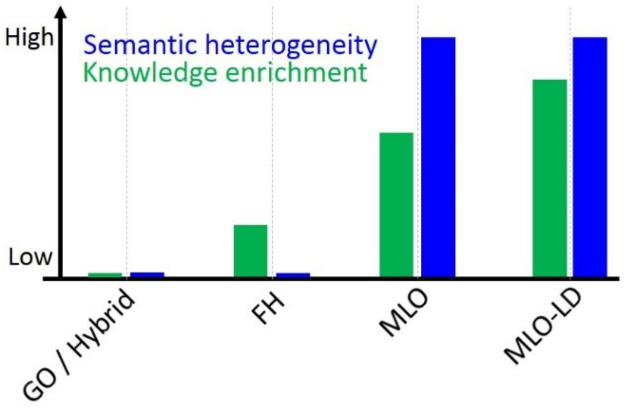


Figure 30 : Knowledge model management approaches expected performances

3. **MLO-LD**
   Each device locally defines and embeds its own domain ontology. But, unlike the MLO approach, GCI can be HTTP URIs (dereferenced and interlinked with LOV). This approach is good at managing the semantic heterogeneity and, while it cannot completely make the economy of an alignment engine (LOV are not aligned), it allows reducing its inaccuracies.

4.8 Experimentations

4.8.1 Use-case#1 : A New Environment Exploration

We consider in this first use-case (Figure 31) the possible moves of an elderly person in her macroscopic environment. 99% of the time, this person is either located at home (yellow circle) or run errands (blue circle). While the person remains inside this cycle (pink cycle), no new device are discovered in her environment and the system knowledge remains stable but potentially incomplete. Then, exceptionally, this person has to visit a friend (green circle). Once in her friend's environment, new devices are discovered contributing at enriching the system knowledge and potentially incrementing the initial incomplete knowledge. Back to the traditional move cycle, the newly added knowledge may leads the system to better assist the person.

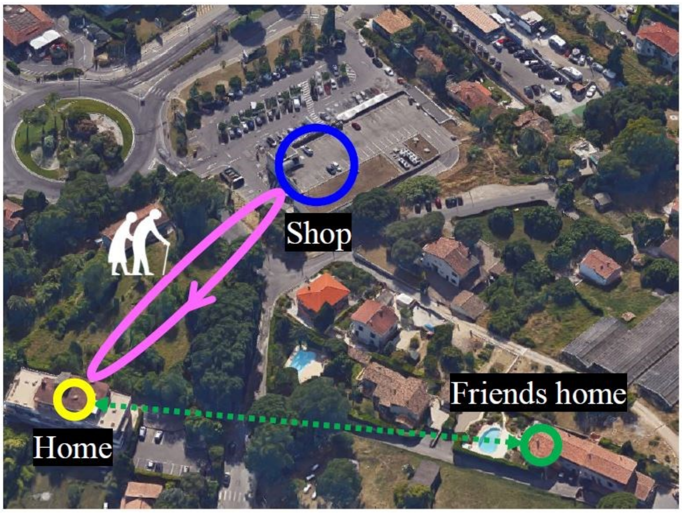


Figure 31 : Elederly people displacement scheme

4.8.2 Use-case#2 : Search For Energy-Efficient Devices

In this case study the system searches for energy-efficient appliances for playing a music track. The environment initially comprises the following appliances: an Android tablet and a hi-fi system installed in the living room. These appliances embed devices allowing them to be monitored and controlled by the system. Devices provide semantic annotations describing: (1) the appliance power consumption (as a data property), (2) some terminological concepts about their domains. The problem occurring in the context of searching for energy-efficient devices instances from the available knowledge is that using the appliance power consumption property and an arbitrary trigger may lead to inaccurately discriminate the devices…

Let's consider now that the inhabitant install a new electric meter in the environment. This electric meter brings new knowledge about the energy classification for home appliances that can be based, for instance, on the European Union energy label[1]. This new knowledge is brought in the form of SWRL rules defined in the device annotations and enriches the knowledge base upon device discovery. The reasoning engine then infers, for each device instance in the knowledge base, a new property defining the European Union energy label from the initial power consumption property. It permits to more efficiently search for device instances based on a parameter making sense in the domain of the energy consumption.

4.8.3 Experimentations Setup

The previously described scenario has been tested using the CONTINUUM platform[2] enhanced thanks to the contribution presented in this paper. WComp middleware [25], for service composition by assembling light components, is at the heart of this platform. It implements the SLCA model (Lightweight Service Component Architecture) [26] where the application is formed with an assembly of software components based on the LCA model (Lightweight Component Architecture) and services communicating using events. A functional interface giving access to the functional services is exported. This platform is based on UPnP (Universal Plug and Play). Like DPWS (Device Profile for Web Services), this protocol allows to dynamically manage devices (discovery and disappearance) and registration to the proposed services. This platform is coupled with

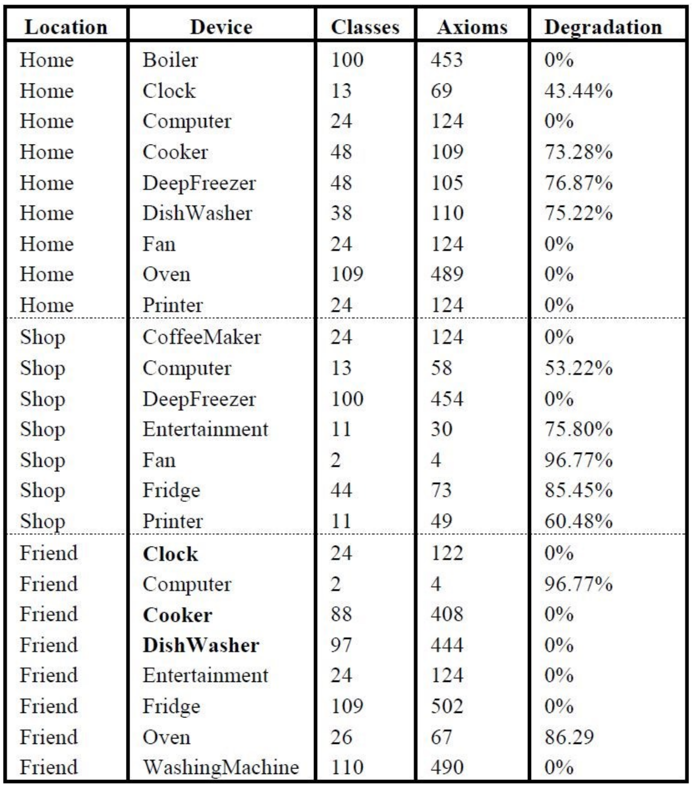
[1] http://en.wikipedia.org/wiki/European˙Union˙energy˙label
[2] Project for service continuity in ubiquitous and mobile computing - French national research agency - ANR-08-VERS-0005.

Conquer knowledge base [27] built on top of Jena API. This knowledge base has been encapsulated in a web service for device (Universal Plug and Play, UPnP) and enhanced with Pellet reasoning engine [28] able to infer on SWRL rules (Table 5), the Alignment API [29] and some real time ontology metrics monitoring capabilities. Using the aforementioned platform, composite web services have been created for each device, exposing an interface allowing the knowledge base to retrieve the semantic metadata upon device discovery. The metadata are written following the RDF/XML format.

4.8.4 Results

a) Use-case#1 : A New Environment Exploration

To the best of our knowledge, there is currently no dataset available on the web applicable to validate the proposed approach. Instead, most of the works are relying on a comprehensive ontology at a basis to describe all the knowledge for a given domain. Since ontology engineering is a time consuming task necessitating expertise to ensure knowledge modeling coherency, we have used DogOnt ontology [30] rev 3.2.11 describing 926 concepts and containing 9383 axioms. This ontology is general enough to be used in a wide range of domains. The dataset is then created by fragmenting the ontology into sub-ontologies defining and structuring all the knowledge necessary to fully describe some devices. Then, from each sub-ontology, are generated a set of degraded sub-ontologies (Figure 33) containing a subset of the device complete knowledge. Using this approach has permitted to elaborate a comprehensive electrical appliances dataset used to get reproducible measures while keeping the control on the fragmentation and degradation rates. From multiple local ontologies approaches standpoint, this experimental dataset assumes that linked data and alignment engine perfectly smooth the semantic heterogeneity appearing when dealing with ontologies independently developed.



Location	Device	Classes	Axioms	Degradation
Home	Boiler	100	453	0%
Home	Clock	13	69	43.44%
Home	Computer	24	124	0%
Home	Cooker	48	109	73.28%
Home	DeepFreezer	48	105	76.87%
Home	DishWasher	38	110	75.22%
Home	Fan	24	124	0%
Home	Oven	109	489	0%
Home	Printer	24	124	0%
Shop	CoffeeMaker	24	124	0%
Shop	Computer	13	58	53.22%
Shop	DeepFreezer	100	454	0%
Shop	Entertainment	11	30	75.80%
Shop	Fan	2	4	96.77%
Shop	Fridge	44	73	85.45%
Shop	Printer	11	49	60.48%
Friend	**Clock**	24	122	0%
Friend	Computer	2	4	96.77%
Friend	**Cooker**	88	408	0%
Friend	**DishWasher**	97	444	0%
Friend	Entertainment	24	124	0%
Friend	Fridge	109	502	0%
Friend	Oven	26	67	86.29
Friend	WashingMachine	110	490	0%

Figure 33 : Use-case#1 local ontologies (potentially incomplete)

Results are exhibited in the Figure 34. After having discovered all devices in the usual environment of the elderly person (1), the system knowledge (blue curve) remains flat as long as the person does not come out of this environment (2). The person visits her friend and new devices are discovered in this new environment (3). The newly added knowledge is made persistent in the system when the person is back to home (4). New knowledge has been added on the clock, the cooker and the dishwasher appliances (Figure 33). This leads the system to potentially improve the relevancy of devices to be used in concert and then better assist the elderly person in her everyday life.

b) Use-case#2 : Search for Energy-efficient Devices

For this use-case, we have developed simple heterogeneous ontologies describing a Hi-fi system and an Android tablet along with a power consumption property (Figure 36 and Figure 35). The electric meter ontology defines SWRL rules allowing to classify the devices based on their power consumption. For instance, the following rule infers that devices with a power consumption property value in between 1W and 10W are classified in category “A”:

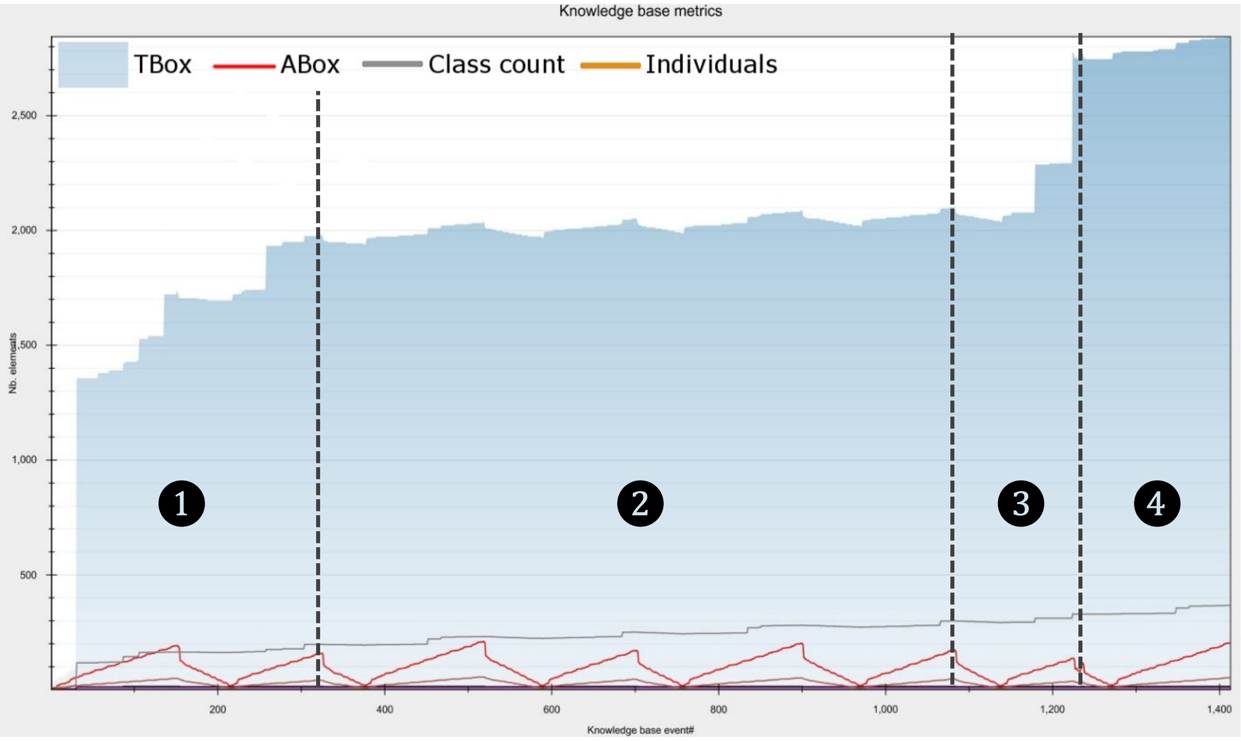


Figure 34 : Use case#1 execution results

Following the use-case described in Paragraph 4.8.2, two devices are first added in the environment: (1) an Android tablet with 8W power consumption, (2) a Hi-fi sound player with 28W power consumption. Those devices are then discovered and their semantic annotations are used to enrich the knowledge base. The alignment engine links “Appliance*” and “Device*” concepts together (`owl:equivalentClass`). We consider that only the Android tablet is relevant to play a music track with the lower power consumption. At this point, a query is executed to retrieve “Speaker” type devices with a power consumption lower than 30 watts (arbitrary chosen value):

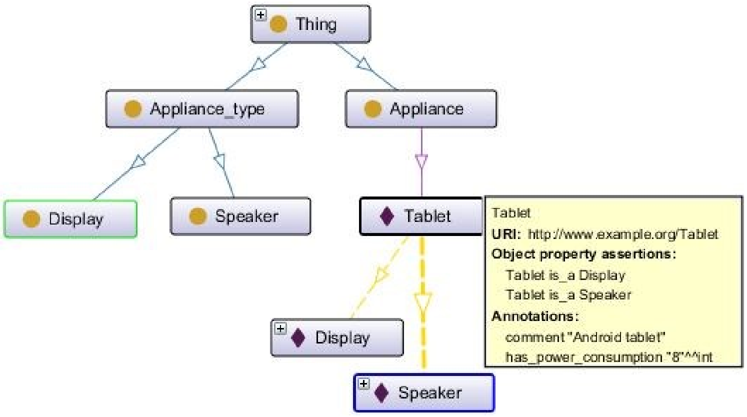
With the previous query, both devices are returned:

An electric counter device is added bringing new knowledge about the energy classification for home appliances that can be based, for instance, on the European Union energy label. This new knowledge is added in the form of SWRL rules. A new query can be executed to show up the inference engine execution results (inferring the property “`has_consumption_category`”):

The newly created property allows to classify the devices power consumption under term and values making sense in the power consumption domain:

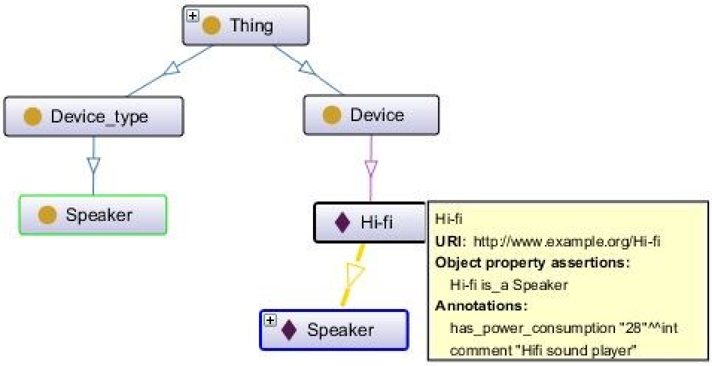
We are now able to slightly modify the previous query into a more relevant one exploiting the newly added property:

Thanks to the added knowledge, the most relevant device is now the only one selected:



Tablet
URI: http://www.example.org/Tablet
Object property assertions:
Tablet is_a Display
Tablet is_a Speaker
Annotations:
comment "Android tablet"
has_power_consumption "8"^^int

Figure 35 : Android tablet simple ontology



Hi-fi
URI: http://www.example.org/Hi-fi
Object property assertions:
Hi-fi is_a Speaker
Annotations:
has_power_consumption "28"^^int
comment "Hifi sound player"

Figure 36 : Hifi simple ontology

4.9 Conclusion

In this chapter, we have briefly introduced the knowledge base concepts and its foundations (Paragraph. 4.2) along with reasoning engines used to infer implicit facts (Paragraph. 4.3). Then, we have depicted three knowledge dynamicity levels (property level, instance level and terminological level) inherent to the physical nature of the environment and the devices along with their impacts on the integration process (Paragraph. 4.4). We have presented some ontology interoperability techniques helping to cope with heterogeneous metal models integration (Paragraph. 4.5). These technics are currently not perfect, but it is still a hot topic having good momentum in the community. On the other side, we demonstrated how heterogeneous meta models are good at increasing the knowledge accuracy over time (Paragraph 4.6), the same terms might be differently described in the several heterogeneous ontologies hereby acquiring a more accurate meaning throughout an evolutionary process, their meaning being shaped through the assertions done by their publishers . We finally proposed a knowledge integration model (Paragraph. 4.7) that makes GCI persistent in the knowledge base. It has been validated on two use-cases (Paragraph 4.8).

4.10 References

[1] Baader, F. (2003). The description logic handbook: theory, implementation, and applications. Cambridge university press.

[2] Horrocks, I. (2005). Owl: A description logic based ontology language. In Logic Programming (pp. 1-4). Springer Berlin Heidelberg.

[3] Krötzsch, M., Simancik, F., & Horrocks, I. (2012). A description logic primer. arXiv preprint arXiv:1201.4089.

[4] Horrocks, I., Patel-Schneider, P. F., & Van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. Web semantics: science, services and agents on the World Wide Web, 1(1), 7-26.

[5] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," 2005. [Online]. Available: http://www.w3.org/Submission/SWRL/.

[6] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web, 5(2), 51-53.

[7] Haarslev, V., Hidde, K., Möller, R., & Wessel, M. (2012). The RacerPro knowledge representation and reasoning system. Semantic Web, 3(3), 267-277.

[8] Tsarkov, D., & Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In Automated reasoning (pp. 292-297). Springer Berlin Heidelberg.

[9] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. (2014). HermiT: an OWL 2 reasoner. Journal of Automated Reasoning, 53(3), 245-269.

[10] Dentler, K., Cornet, R., ten Teije, A. C. M., & de Keizer, N. F. (2011). Comparison of reasoners for large ontologies in the OWL 2 EL profile.

[11] Hourdin, V., Tigli, J. Y., Lavirotte, S., Rey, G., & Riveill, M. (2008). SLCA, Composite Services for Ubiquitous Computing. In 5th International Conference on Mobile Technology, Applications and Systems (Mobility'08) (p. 8).

[12] Mayer, S., & Guinard, D. (2011, June). An extensible discovery service for smart things. In Proceedings of the Second International Workshop on Web of Things (p. 7). ACM.

[13] Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D. (2010). Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. Services Computing, IEEE Transactions on, 3(3), 223-235.

[14] Richard, G. G. (2000). Service advertisement and discovery: enabling universal device cooperation. Internet Computing, IEEE, 4(5), 18-26.

[15] Choi, N., Song, I. Y., & Han, H. (2006). A survey on ontology mapping. ACM Sigmod Record, 35(3), 34-41.

[16] Tulasi, R. L., & Rao, M. S. (2014). Survey on Techniques for Ontology Interoperability in Semantic Web. Global Journal of Computer Science and Technology, 14(2).

[17] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001, August). Ontology-based integration of information-a survey of existing approaches. In IJCAI-01 workshop: ontologies and information sharing (Vol. 2001, pp. 108-117).

[18] Shvaiko, P., & Euzenat, J. (2013). Ontology matching: state of the art and future challenges. Knowledge and Data Engineering, IEEE Transactions on, 25(1), 158-176.

[19] Ehrig, M., & Staab, S. (2004). QOM–quick ontology mapping. In The Semantic Web–ISWC 2004 (pp. 683-697). Springer Berlin Heidelberg.

[20] Euzenat, J., & Shvaiko, P. (2013). Classifications of ontology matching techniques. In Ontology matching (pp. 73-84). Springer Berlin Heidelberg.

[21] Dragisic, Z., Eckert, K., Euzenat, J., Faria, D., Ferrara, A., Granada, R., ... & Grau, B. C. (2014, October). Results of the ontology alignment evaluation initiative 2014. In Proceedings of the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference (ISWC 2014).

[22] Goumopoulos, Christos, et al. "Atraco: Adaptive and trusted ambient ecologies." Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on. IEEE, 2008.

[23] Sabou, M., Kantorovitch, J., Nikolov, A., Tokmakoff, A., Zhou, X., & Motta, E. (2009). Position paper on realizing smart products: Challenges for semantic web technologies. In CEUR Workshop Proceedings (Vol. 522, pp. 135-147).

[24] Hartmann, M., Uren, V., & Vildjiounaite, E. Gathering knowledge for supporting interaction with smart products.

[25] Tigli, J. Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., & Riveill, M. (2009). WComp middleware for ubiquitous computing: Aspects and composite event-based Web services. annals of telecommunications-annales des télécommunications, 64(3-4), 197-214.

[26] Hourdin, V., Tigli, J. Y., Lavirotte, S., Rey, G., & Riveill, M. (2008). SLCA, Composite Services for Ubiquitous Computing. In 5th International Conference on Mobile Technology, Applications and Systems (Mobility'08)

[27] Benyelloul, A., Jouanot, F., & Rousset, M. C. (2010). Conquer, an RDFS-based model for context querying. In 6emes Journées Francophones Mobilité et Ubiquité.

[28] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web, 5(2), 51-53.

[29] David, J., Euzenat, J., Scharffe, F., & Dos Santos, C. T. (2011). The alignment api 4.0. Semantic web journal, 2(1), 3-10.

[30] Bonino, D., & Corno, F. (2008). Dogont-ontology modeling for intelligent domotic environments (pp. 790-803). Springer Berlin Heidelberg.

[31] Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. Communications Surveys & Tutorials, IEEE, 16(1), 414-454.

[32] Aggarwal, C. C., Ashish, N., & Sheth, A. P. (2013). The Internet of Things: A Survey from the Data-Centric Perspective.

5. Knowledge management

5.1 Introduction

The main idea developed in Chapter. 4 is about permitting the knowledge base GCI to be enriched throughout the life of the system hereby permitting described terms to acquire a more accurate meaning over time. The persistence of this knowledge (Paragraph. 4.7) brings two potential issues from a knowledge base management standpoint: (1) it is unlikely that the knowledge base GCI content can indefinitely grow. Considering the ambient systems potential low available computational resources, limitations may occur in space (system memory limitation) and time (e.g. reasoning processing time). A trade-off has to be found in between permitting the knowledge base GCI enrichment, the intrinsic system capabilities (CPU, memory) and the user experience (system responsiveness); (2) In the context of knowledge base GCI content enrichment over time along with devices and services whose functionalities can be upgraded remotely by the manufacturers, it is important for the knowledge base to keep its GCI content *at the highest quality* level over time (up to date, consistent and coherent). Therefore, the challenge for the ambient system is about managing the GCI *lifecycle* without degrading the knowledge enrichment capacity at improving the relevancy of the selected devices and services.

The aforementioned knowledge management issues are not addressed in this document but we give a short overview of possible ways to overcome them. In the first part of this chapter (Paragraph. 5.2) we review some algorithms that can be put in place whose role is to permit the ambient system to remove GCI data from the knowledge base without losing the meaning of the terms learnt throughout the life of the system. In the second part, we review some ways permitting the ambient system to deal with corrupted or obsolete GCI data over time and accommodate change (Paragraph. 5.3).

5.2 GCI Size Containment

GCI size containment is about, for the system, when a critical size is reached, to remove GCI from the knowledge base. The main challenge is that these removal must not deteriorate the relevancy (See Paragraph. 6.3) of the further selected devices and services (e.g. increasing false positive or false negative). We present here after some metrics and algorithms that can be used for that purpose.

Authors in [5][7] denotes several metrics for knowledge base structural analysis. For the purpose of identifying GCI to keep (or remove), the following metrics might be valuable:

1. *Class usage* giving the classes are the most frequently instantiated (rdf:type);
2. *Triples per subject class*. Understand how triples are distributed over subject classes;
3. *Property usage*. Get the most frequently used properties;
4. *Property Usage per Subject Class*. Get the most frequent combinations of subject class and property;
5. Etc…

On the other side, some metrics for the knowledge base usage analysis might be measured. Typically, a concept description model might have been integrated once and never instantiated any more. Basically, we refer here to some memory management mechanisms (Least Recently Used (LRU), garbage collection, etc…) all based to the fact that some memory slots may have been allocated but are not used or reachable anymore and can be removed to free up some memory space. In the context of knowledge base (and more generally in the context of graph based modeling), several concepts might never be instantiated but are part of a more global knowledge (by subsumption).

An approach would be to clustered the knowledge base content (one cluster per concept) and give each cluster a “popularity rate” value (a cluster can be seen as a sub-ontology describing a concept).

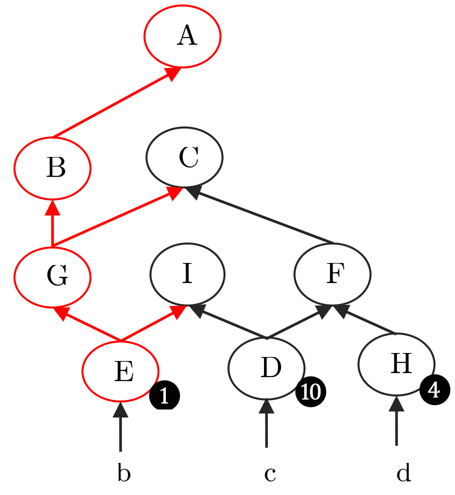


Figure 37 : Ontology exemple : E concept removal

Given an ontology O, its set of concepts $C_O = \{c_1, c_2, \dots, c_x\}$, x ∈ ℕ, its set of properties $P_O = \{p_1, p_2, \dots, p_y\}$, y ∈ ℕ, and its set of instances I_o=$\{i_1, i_2, \dots, i_z\}$, z ∈ ℕ. Given the set of its instantiated class I_c=$\{I_1, I_2, \dots, I_n\}$, n ∈ ℕ, and their associated sub-ontology C_{I_c} = $\{C_{I_1}, C_{I_2}, \dots, C_{I_m}\}$, p ∈ ℕ.

<u>A concept μ in C_{I_j} can be removed if:</u>

1) for $k = \{1, \dots, m\} \neq j, C_{I_j} \cap\ C_{I_k} \neq\ \mu,$

2) for $k = \{1, \dots, x\}, c_k \neq \mu,\ domain(p(c_k)) \neq\ \mu$ and r$ange(p(c_k)) \neq\ \mu$

3) for $k = \{1, \dots, z\}, i_k \neq \mu, domain(p_{i_k}) \neq\ \mu$ and $range(p_{i_k}) \neq\ \mu$

Example: let’s consider the ontology depicted in Figure 37. The concept E has been instantiated only once throughout the life of the system and it is decided to remove it from the knowledge base.

C_{I_1} = {I, F, C}	(D)	$C_{I_2} \cap C_{I_1}$= {I, C}
C_{I_2} = {I, G, C, B, A}	(E)	
C_{I_3} = {F}	(H)	$C_{I_2} \cap C_{I_3}$= {∅}

Concepts I and C cannot be removed. But, concepts G,B and A can be removed along with the concept E (considering that none of the concepts nor instances have properties using E as range or domain).

5.3 GCI Quality Over Time

5.3.1 Incoherencies and Inconsistencies Management

Knowledge base content inconsistency and incoherency terms are defined in Paragraph 4.3.2. An illustration is depicted in Figure 38 [1].

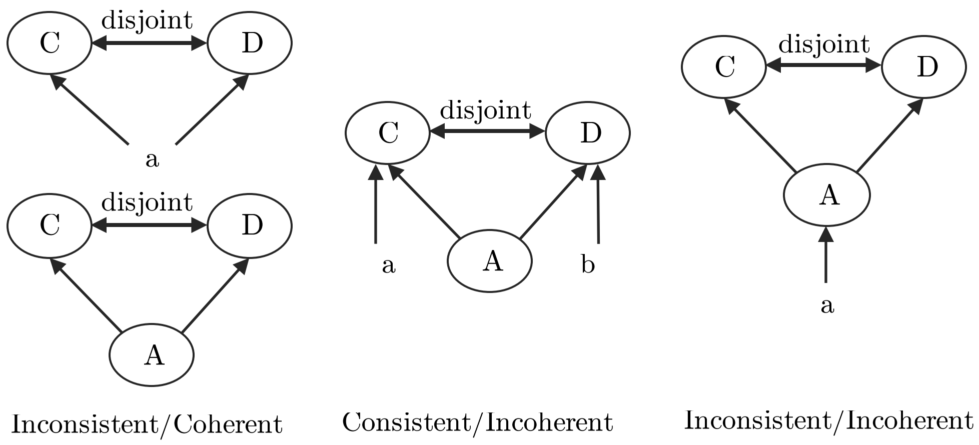


Figure 38 : Ontology incoherency and inconsistency examples

Such anomalies can be automatically detected and are either:

1. Intrinsic to an ontology due to design errors (misinterpretation of the terms meaning, etc…). Automatic *repair* is complex and has to be performed manually by domain experts [2]. Some intermediate solutions can be implemented. For instance, the system might be configured to not integrate GCI leading to knowledge base inconsistencies and/or incoherencies,
2. Produced by alignments, mapping and/or integration process [3] (Paragraph. 4.5). Automatic repair in that case is, for example, about removing mappings that lead to unsatisfiable classes in order to ensure that the final alignment is coherent [4].

Note that GCI might be erroneous but not necessarily leading to incoherencies and inconsistencies. The challenge for the system is to detect such erroneous data. An approach would be to measure the incoming GCI similarity with the already integrated knowledge base content defining the same GCI (See Paragraph. 6.3.2). Based on a similarity threshold value the system may decide to integrate or not the incoming GCI. However, in the context of the proposed MLO approach and its underlying capability at enriching the knowledge base over time and give terms a more accurate meaning, a given term might be used in distinct contexts and similarity measure might be low. Thus the system may decide, based on this low similarity measure, to not integrate the incoming GCI even though it would have been worth to do so in order to enrich the term meaning.

5.3.2 Validity

Since the knowledge base GCI content is enriched over time and because devices and services functionalities can be upgraded remotely by the manufacturers, some GCI in the knowledge base might become obsolete over time and have to be updated. The main issue the system has to cope with is to decide either if:

1) **An incoming GCI is a new revision of a previously integrated one and has to replace it**; For the system to be able to manage such a situation, it would require a time stamp or a revision number to be added:
   a) In the device metadata. The system then get the metadata GCI content and keep track of the time stamp or revision number value along with the URI prefix (the publisher) and GCI description identifier. Any new device bringing the same GCI description identifier and URI prefix with an higher revision number would trig the knowledge base update,
   b) In case of dereferenceable URIs, the associated RDF graph might integrate a revision number as well.
2) **An integrated GCI is still up to date**. Here we denote three possible scenarios:
   a) The device publishing the GCI remains permanently in the system environment (Figure. 19). As this device metadata can be remotely upgraded by the manufacturer, the system has to perform a device refresh request (ask the device to emit its metadata) in a daily or weekly basis,
   b) The device model GCI description is gathered from dereferenceable URI. Here again the system, on a daily or a weekly basis, can dereference the URI to obtain the graph and check for its revision,
   c) The device publishing the GCI is no more available in the system environment. The potential needed update might not be available until the device is rediscovered in the environment.

For all cases, validity information has to be given by the publishers (manufacturers) not the consumers (the ambient systems).

5.4 Conclusion

We have presented in this chapter a short overview of the possible solutions that can be put in place to contain the system knowledge base size and manage its content quality over time. In order to relieve the system and increase its responsiveness, knowledge base maintenance like validity checking (Paragraph. 5.3.2) might be executed on a daily or a weekly basis. Knowledge base incoherency and inconsistency management (Paragraph. 5.3.1) has to be done before new incoming GCI integration.

5.5 References

[1] Flouris, G., Huang, Z., Pan, J. Z., Plexousakis, D., & Wache, H. (2006, July). Inconsistencies, negations and changes in ontologies. In Proceedings of the National Conference on Artificial Intelligence (Vol. 21, No. 2, p. 1295).

[2] Gkaniatsou, A., Bundy, A., & Mcneill, F. (2012, November). Towards the automatic detection and correction of errors in automatically constructed ontologies. In Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on (pp. 860-867). IEEE.

[3] Zhang, S., Grau, B. C., & Jimenez-Ruiz, E. (2014). Inconsistency Repair in Ontology Matching (Doctoral dissertation, MSc thesis., University of Oxford).

[4] Pesquita, C., Faria, D., Santos, E., & Couto, F. M. (2013, October). To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In OM (pp. 13-24).

[5] Holst, T. (2013). Structural analysis of unknown RDF datasets via SPARQL endpoints (Doctoral dissertation, Master thesis defense 11)

[6] Scott, J. (2012). Social network analysis. Sage.

[7] Peroni, S., Motta, E., & d'Aquin, M. (2008). Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In The Semantic Web (pp. 242-256). Springer Berlin Heidelberg.

6. Knowledge retrieval

6.1 Introduction

So far we do have a knowledge base: (1) whose GCI is enriched throughout the life of the system hereby permitting described terms to acquire a more accurate meaning over time, (2) whose size is under control and its content *valid*, (3) which is coherent with its context, at each instant, it only contains the instances of the available devices in the environment. It is now time for the ambient system, based on this knowledge, to retrieve the most *relevant* devices and services instances to make them work in concert to assist users. The devices and services selection mechanism is based on selection rules developed at design time. Usually, the knowledge base data retrieval (e.g. from selection rules) is achieved using SPARQL (SPARQL Protocol and RDF Query Language)[1], queries being expressed using GCI defined in the knowledge base.

However, based on the proposed approach, the knowledge base content, at design time, is unknown (possibly empty) both at the instances and the GCI levels. The problem is then twofold : (1) find a way to express rules at design time independently of the knowledge base GCI content; (2) the knowledge base content being unknown, rules cannot be tuned at design time to increase the recall and, the system having to work autonomously (without user interactions), rules cannot be updated or tuned at run time neither. Even so, the retrieved devices and services instances have to be the most relevant ones according to the defined rules and taking into account the knowledge base GCI content enrichment over time.

In this chapter, after having reviewed the SPARQL query language in Paragraph. 6.2 and its limitations with regards to the proposed approach, we propose, in the Paragraph. 6.3, a selection rule model independent of the knowledge base content and define in Paragraph 6.3.2, a criteria based on ontology similarity measure, aimed at giving the system an insight, at run time, on the relevancy of the selected devices and instances versus the associated selection rule. Finally, the model is validated on a use-case and results presented in Paragraph. 6.4.

6.2 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [1][2] is a query language developed primarily to query RDF graphs. A SPARQL endpoint provides the users and applications a mechanism to retrieve information from a knowledge base using SPARQL queries. SPARQL queries are composed by a set of RDF triple patterns (Basic Graph Patterns (BGP)), in which variables might appear. Complex SPARQL queries can be elaborated by using union (UNION operator), constraints (FILTER operator), etc… The output of a SPARQL query can be of different types: yes/no answer (ASK), selections of values of the variables which match the patterns (SELECT), construction of new triples from these values (CONSTRUCT), and, from SPARQL 1.1, the possibility to add/remove and modify RDF queried data (UPDATE).
The structure of a SPARQL query is depicted here after:

For instance, hereafter are depicted two queries examples. The first query below retrieves all devices instances and print out the results in an array (output format can be either XML, JSON, CSV, etc...) where the first column is the extracted concepts stored in the variable ?concept, and the second column is the associated instances stored in the variable ?instance.

?instance
TV
IPad

The second query retrieves all elements for which the property core:has˙consumption˙category holds and print out the results in an array where the first column is the element retrieved stored in the variable ?element and the second column is the value associated and stored in the variable ?label.

?element	**?label**
WashingMachine	“A”
TV	“D”

The main mechanism for computing query results in SPARQL is subgraph matching [3]: RDF triples in the queried RDF data and the query pattern (BGP set) are interpreted as directed graphs, and matched together (simple entailment) using query variables as wild cards (the underlying algorithm tries to find all subgraphs in the data graph matching with the query RDF graph. Variables are replaced with values found in the matching sub-graph and returned as result). For instance, the first query example is interpreted as the RDF graph depicted in the Figure 39.

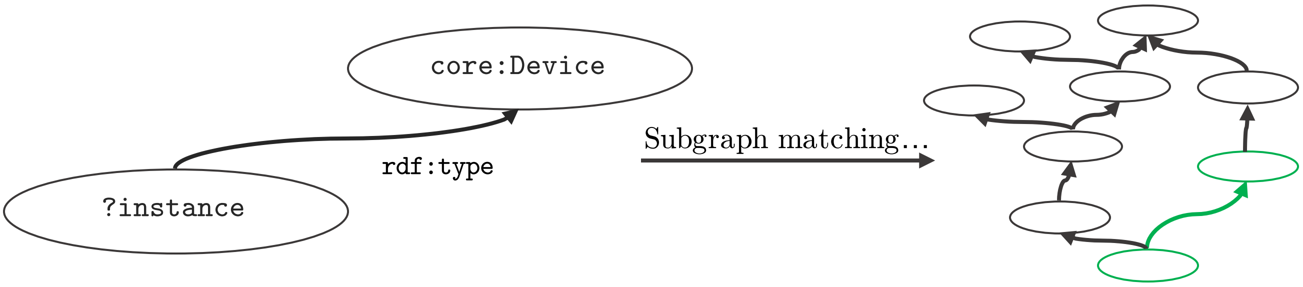


Fig. 39.a : Query

Fig. 39.b : Knowledge base

Figure 39 : SPARQL query RDF subgraph matching

[1] http://www.w3.org/TR/2009/WD-sparql11-entailment-20091022/

SPARQL 1.1 [2] introduces, among other features, entailment regimes[1] (RDFS, OWL family) to allow for finding *additional* answers to a query that are not directly specified in the queried graph, but can be *inferred* using a set of inference rules (entailment regimes supported are SPARQL engine implementation dependent). **`Entailment regimes are good at working with knowledge base GCI enriched over time`**.

However, Expressive queries (e.g. the second query previously described) requires to know explicitly the namespaces (URI prefixes) used in the knowledge base (e.g. `PREFIX core: <http://www.example.org/elecm#>`) and the GCI (e.g. `core:has_consumption_category`). Note that an approach based on regular expressions might be used as well to filter the results but would be purely syntactic.

So, SPARQL query based rules are powerful when the queried knowledge base GCI content is known, but are inapplicable (See also the paragraph 6.3.2) if the knowledge base GCI content is unknown (assuming expressive BGP and not generic ones (e.g. `?i rdf:type ?c`)). With the MLO approach proposed (Paragraph 2.6.2), since the devices and services description models are based on heterogeneous and unknown meta models at design time, selection rules cannot be based on SPARQL queries (which are modeled based on top of the meta model defined in the knowledge base GCI).

6.3 Selection Rule Model Proposal

As seen in the previous paragraph: (1) thanks to BGP, SPARQL queries describe the *explicit* semantics of the searched elements (meaning that the query description model relies on a meta model defined by the knowledge base GCI) and therefore, one is sure that answers returned are semantically matching to the query; (2) thanks to entailment regimes, additional answers to a query that are not directly specified in the queried graph can be inferred using a set of inference rules.

Like SPARQL where an RDF graph describing a model of a query (Figure 41) relying on the meta model defined by the knowledge base GCI, we propose an approach based on a model to describe a rule but based on its own meta model (Figure 42). The advantage of the proposed approach is that the rule expressivity (up to OWL 2 DL[1] ($\mathcal{SROIQ}$)) is much higher than SPARQL based queries (RDF) and is independent of the knowledge base meta model. Continuing the parallel of the proposed approach with SPARQL, we can also apply a subgraph matching where queried knowledge base content is matched with the rule ontology content [10].

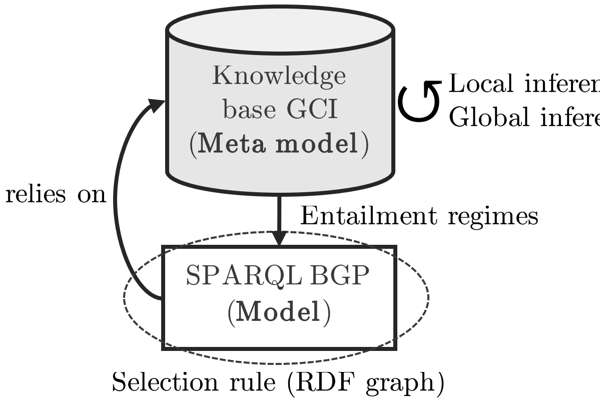


Figure 41 : SPARQL query based rule model

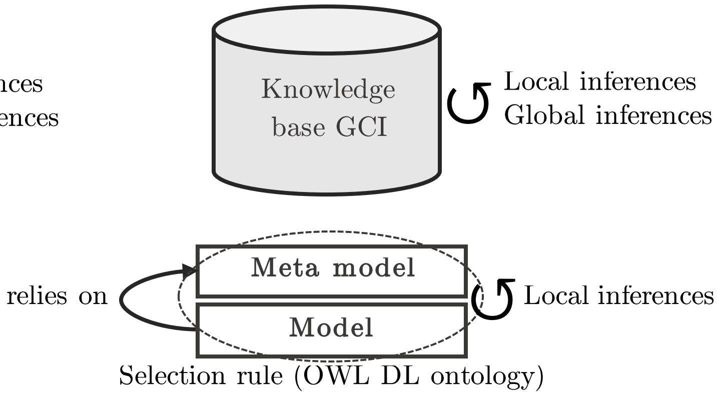


Figure 40 : Proposed rule model

[1] Note that rules description models can either be written down from scratch or using tools to, for instance, convert natural language (NL) sentences to RDF graphs or OWL [5][6].

However: (1) in order for the subgraph matching algorithm to be efficient (the rules meta models (rules GCI) being heterogeneous from the knowledge base GCI content standpoint), some challenges have to be overcame; (2) we need a SPARQL entailment regimes equivalent mechanism for rules defined at design time to lead more relevant results at run time according to the knowledge base GCI content evolution over time.

6.3.1 Selection Rule Semantic Heterogeneity Management

Selection rules GCI have first to be aligned with knowledge base GCI. Ontology alignment mechanism is presented in the Paragraph. 4.5.2. In a nutshell, ontology *alignment* process takes two input ontologies and produces a set of correspondences between concepts that match *semantically* with each other (Ontology matching is the process of discovering similarities ($\in \mathbb{R}$) between two ontologies). These matches are also called mappings [7].

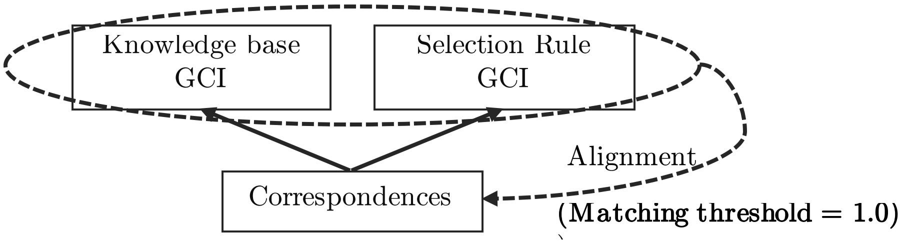


Figure 42 : Selection Rule GCI alignment with knowledge base GCI

Once the alignment process has been executed, the correspondences set has to be integrated in the selection rule model. Here again, we do recommend usage of Linked Open Vocabularies (LOV) to define the selection rule meta models.

6.3.2 From Subgraphs Matching to Ontology Similarity Measurement

Let's consider a graph $G = (V, E, T)$ where $T: V \to \sum^*$ is a labeling function that assigns a label to each vertex in V, and the subgraph query $q = \left(V_q, E_q, T_q\right)$ in G where $T_q: V \to \sum^*$ represents the label constraint for each vertex in V_q. The subgraph matching problem definition is the following [9]:
For a graph G and a subgraph query q, the goal of subgraph matching is to find every subgraph $g = \left(V_g, E_g\right)$ in G such that there exists a bijection $f: V_q \to V_g$ that satisfies $\forall v \in V_q, T_q(v) = T_G(f(v))$ and $\forall e = (u, v) \in E_{q,}\ (f(u), f(v)) \in E_G$ where $T_G(f(v))$ represents the label of the vertex $f(v)$ in G.
This implies V, E, V_q and E_q to use identical labels (URIs). V and E being unknown (MLO approach) there is no chance that V_q and E_q use the same label leading unpractical the subgraph matching approach.

To cope with this issue, we propose to compare semantically G and q. Such a comparison can be achieved by measuring the similarity ($[0, \dots, 1] \in \mathbb{R}$) between G and q [12][13].

<u>It could be ontology distances measures based on [13] (so called similarity in the ontology space):</u>

1. Lexical measures based on Vector Space Model (VSM)[1]
    a. Hamming distance,
    b. Jaccard index[2],
    c. Cosine index[3],
2. Distance between ontology entities
    a. Label based distance (Lexical aggregation-based similarity measure),
    b. Structural distance (OLA similarity [15], Triple-based iterative [13]))
    c. Collection distance (linkage criteria[5], Hausdorff distance[4], Minimum weight maximum graph matching distance [13]).

[1] https://en.wikipedia.org/wiki/Vector˙space˙model
[2] https://en.wikipedia.org/wiki/Jaccard˙index
[3] https://en.wikipedia.org/wiki/Cosine˙similarity
[4] https://en.wikipedia.org/wiki/Hausdorff distance
[5] https://en.wikipedia.org/wiki/Hierarchical˙clustering

The overall similarity measure being partly based on lexical and structural measures, its value depends on the alignment threshold discussed in Paragraph. 4.5.2. The lower is the threshold, the higher might be the similarity measure…

One of the problem we have to cope with is that the queried knowledge base is (over time) much bigger than the selection rule ontology and then, there is few chance that similarity measures will give good results. The idea is to extract, from the knowledge base O, the sub ontology O' containing terms defined in the selection rule. Thus the similarity measures is done on O' and q (sim_{Oq}). For that purpose, the algorithm described in Paragraph. 3.2.2 is extended (Algorithm 2) to incorporate concepts instances in the extracted sub-ontology. This algorithm is repeated for each term defined in the selection rule. Thus, considering $q = \{t_1, t_2, \dots, t_m\}$ the set of terms defined in the query, $O' = \{O'_{t1} \cup O'_{t2} \cup \dots \cup O'_{tn}\}$, n ≤ m. Like for alignment, a similarity threshold has to be chosen (See Paragraph. 6.3.3)

Algorithm 2: Module extraction with instances.

```
begin
    Visited := {};
    R := getResource($S$);
    sList := $O$.listStatements() where statement's subject is equal or equivalent to R;
    for all stmt ∈ sList do
        $O'$ := $O'$ + stmt;
        extract(stmt);
    end for
    iList := R.getInstances();
    for all inst ∈ iList do
        $O'$ := $O'$ + createInstance(inst);
        extract(inst);
    end for
end;

: extract(stmt)
        if(isResource(stmt.subject) and Visited[stmt.subject] = false) {
           Visited[stmt.subject] := true;
           r := getResource(stmt.subject);
           pList := getProperties(r);
           for all p ∈ pList {
               $O'$ += p;
               extract(p);
           end for
        end if
}
```

This approach is quite strict and may lead to low similarity measures. An alternative would be to measure the similarity between the sub-ontologies summaries rather than between the complete sub-ontologies [16][17][18]. Ontology summarization is a useful technique to facilitate ontology understanding and can be expressed as an RDF graph. In the timeframe of this document we did not have room for evaluating such approach.

6.3.3 Selection Rule Answers Ranking

So far, we do have a similarity measure between the selection rule q and the ontology O', extracted from O and containing terms defined in q along with their instances. We need a method to rank the search results (the instances). In other words, we need to compute the relevance of the returned instances. To do that, for each term $t_i \in O'$, its associated sub-ontology $\{O'_{t1}, O'_{t2}, \dots, O'_{tn}\}$ is used to compute the ratio (∈ ℝ) between the amount of terms defined in O'_{t_i} (S_{t_i}) common with terms defined in q over the amount of terms in q (N_q). Each instance of t_i is assigned the value of the ratio, called relevancy measure. This measure is weighted by sim_{Oq} to take into account the semantic distance between the selection rule and the knowledge base content.

$$\boxed{Relevancy_{(Instance_{t_i})} = sim_{Oq} * \frac{S_{t_i} \cap N_q}{N_q}}$$ *Equation 1*

6.3.4 Selection Rule Expressivity *vs* Knowledge Base Content Evolution

The selection rule ontology is defined at design time and, since the system is supposed to work autonomously, the rule is not modifiable nor tunable at run time. On the other side, the knowledge base GCI content is enriched over time and therefore the dissimilarity with the selection rule ontology may increase over time (a dissimilarity is a real positive function *d* of two ontologies which is as large as ontologies differ [13]). We propose, as we do for the knowledge base, to *merge* GCI brought by newly discovered devices with the selection rule ontology. Like with the knowledge base GCI content, the selection rule GCI content is enriched over time, leading to a higher expressivity. And the higher is the expressivity the higher is the selected instances relevancy.

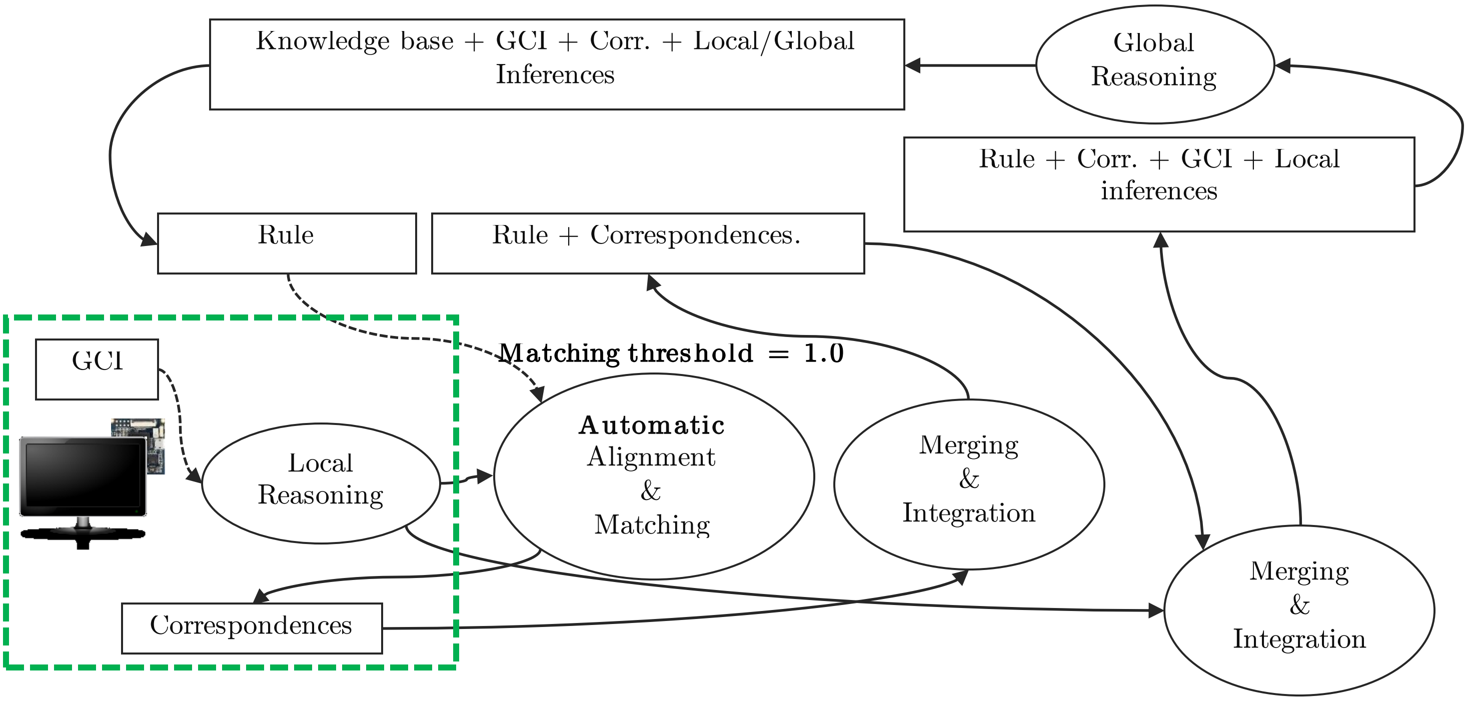


Figure 43 : Selection rule enrichment over time

6.4 Use-case

6.4.1 Selection Rule Definition at Design Time

Let's consider the following rule: "**Television located in the living room**". This rule leads to get the graph depicted in the Figure 44.a. LOV and WordNet equivalent classes might be added (Figure 44.b).

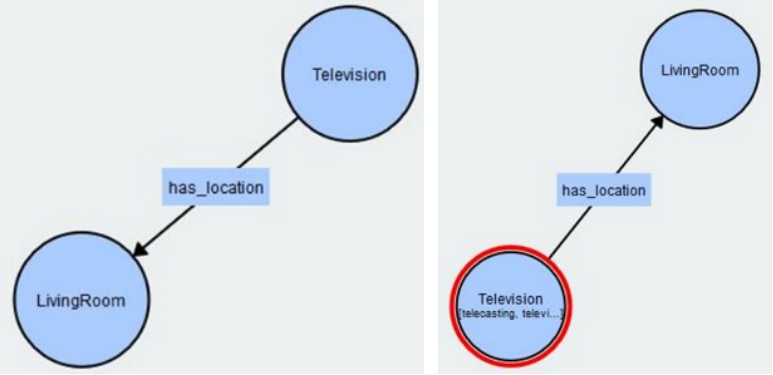


Fig. 44.a *Fig. 44.b*

Figure 44 : Selection rule defined at design time

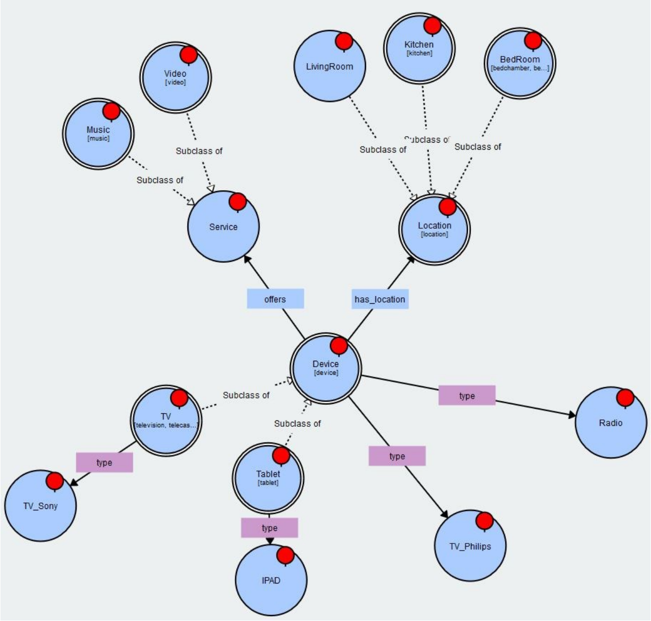


Figure 45 : Knowledge base at t+n

6.4.2 Selection Rule Alignment with Knowledge Base at Run Time

Throughout the life of the system, the knowledge base content is enriched (Figure 45) and the selection rule content is kept aligned with it (Figure 46).

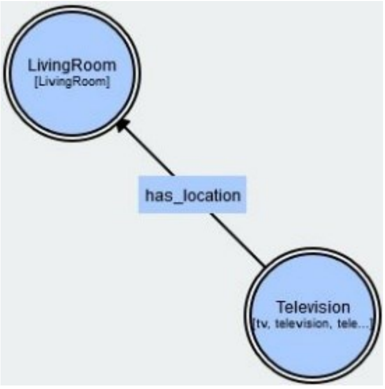


Figure 46 : Selection rule aligned with knowledge base content

6.4.3 Selection Rule Enrichment with New GCI at Run Time

A new device (a basic TV ontology) is discovered in the environment (Figure 47). This newly discovered device brings an ontology that is merged with the selection rule (Figure 48). Doing so the expressivity of the selection rule increases over time.

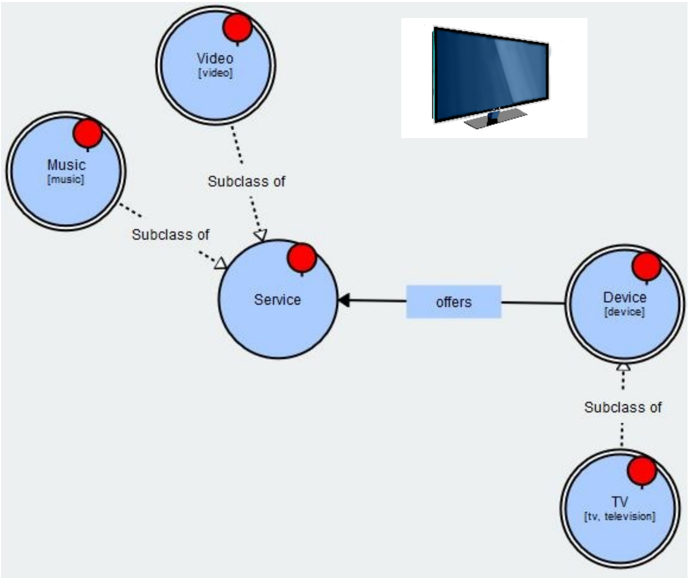


Figure 47 : Basic TV ontology brought by a newly discovered device

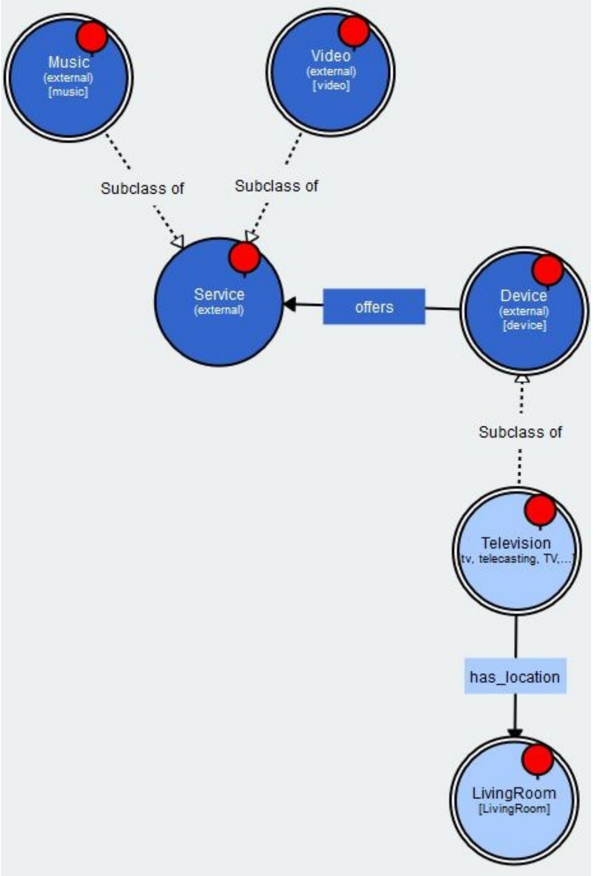


Figure 48 : Basic TV ontology is merged with the selection rule

6.4.4 Selection Rule ⇔ Knowledge Base Similarity Measurement

At this point, a sub-ontology containing terms defined in the selection rule is extracted from the knowledge base (Paragraph. 6.3.2). The sub-ontology extracted is depicted in the Figure 49.
The OntoSim API[1] has been used for measuring the similarity between the selection rule ontology and the extracted sub-ontology. The API is dedicated to the computation of similarities between ontologies and implements algorithms described in Paragraph 6.3.2.

Using Hausdorff distance based algorithm, the similarity measure gives 73.87%. (We have not been able so far (lack of API documentation) to perform measurements with other algorithms such as structural distance, supposed to give better results [13]). Anyway, for the purpose of the experiment it is not critical.

6.4.5 Selection Rule Answers Extraction and Ranking

The idea here is to get instances associated to each concept defined in the sub-ontology and rank them based on Equation. 1 (Paragraph. 6.3.3). We extract, from the knowledge base, the sub-ontology defining the concept TV along with its instances (Algorithm. 2, Paragraph. 6.3.2). The extracted sub-ontology contains one instance named `TV_Sony` (Figure 50)). Let's compute the relevancy of this instance with regards to the selection rule. To do so we compute the ratio ($\in \mathbb{R}$) between the amount of terms defined in the sub-ontology (Figure 49) common with the terms defined in the selection rule over the amount of terms in the selection rule (Figure 48). The sub-ontology has 6 concepts over 7 defined in the selection rule (`Video, Music, Service, Device, LivingRoom` and `TV`). Thus the relevance of `TV_Sony` with regards to the selection rule is given by (Equation. 1):

$$sim_{Oq} * \frac{S_{t_i} \cap N_q}{N_q} = 0.7387 * \frac{6}{7} = 63.31\%$$

[1] http://www.w3.org/2001/sw/wiki/OntoSim

The same approach is repeated for the concept Device (One can extend the approach to the concepts `Service`, `Location`, etc… but for the purpose of the experiment it is not necessary). The overall results are given in the Table 6.

Instance	sim_{Oq}	$\frac{S_{t_i} \cap N_q}{N_q}$	Instance relevancy
`TV_Sony`	73.87%	85.71%	63.31%
`IPad`	73.87%	57.14%	42.21%
`Radio`	73.87%	57.14%	42.21%
`TV_Philips`	73.87%	14.28%	10.55%

Table 6 : Instances ranking with regards to the selection rule

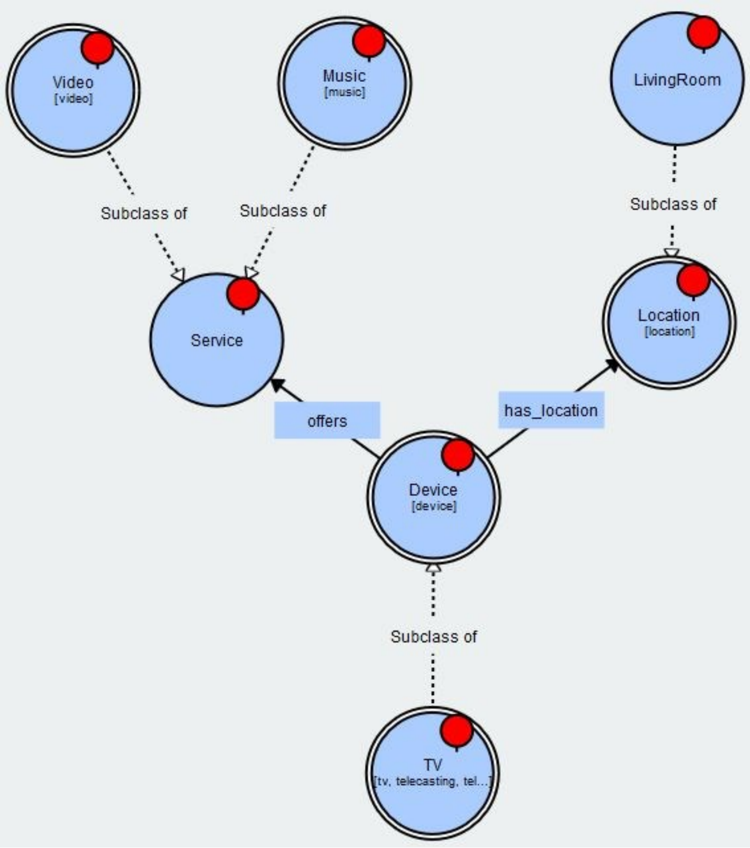


*Figure 49 : Knowledge base extracted sub-ontology **from selection rule***

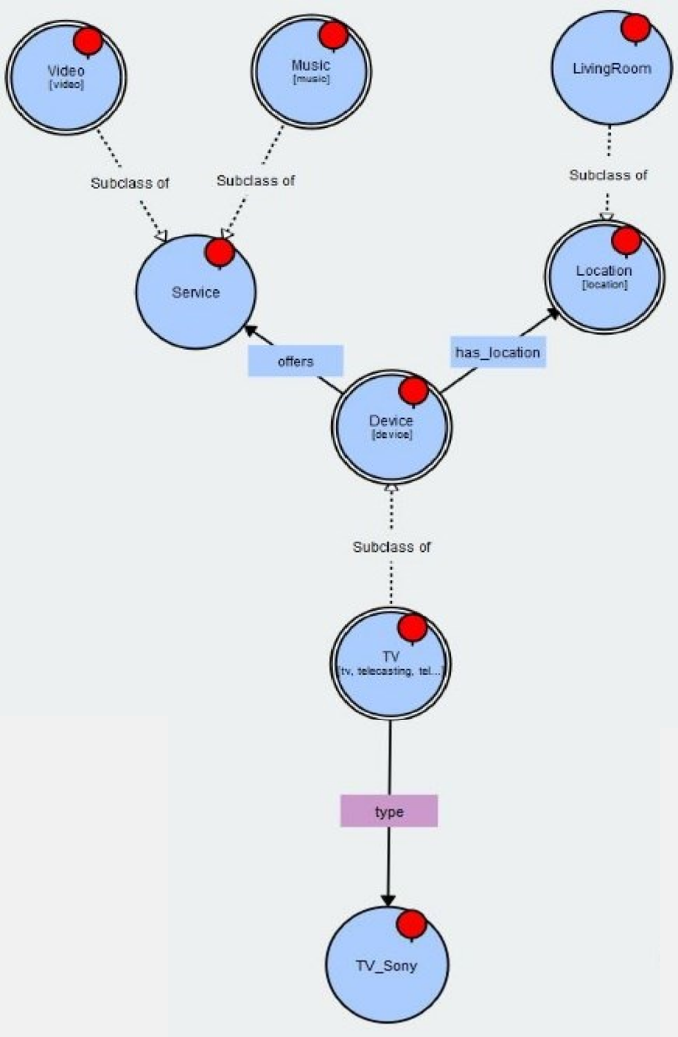


Figure 50 : Sub-ontology defining the concept TV and its instance(s)

6.5 Conclusion

The MLO/MLO-LD approaches brings new issues with regards to knowledge retrieval. Indeed, the knowledge base content is unknown at design and at run time while the system, based on selection rules defined at design time has to return the most relevant instances of the devices and the services. SPARQL query language is typically used to retrieve data from a knowledge base. However, as seen in Paragraph. 6.2, the subgraph matching algorithm SPARQL is based on is not suitable to expressively query unknown content as the meta model the query is built upon is the one defined in the knowledge base GCI (which is unknown). To cope with this issue we have proposed a selection rule model using its own meta model to define the model of the rule (Paragraph. 6.3). Actually, rules are independent OWL ontologies. Thus we have to cope with the semantic heterogeneity between the selection rules ontology content and the knowledge base content but solutions to this problem have been given in the chapter 4 (ontology merge, alignment,…).

Based on this model, instances retrieving is then achieved in two steps: (1) a sub-ontology containing the concepts defined in the selection rule, is extracted from the knowledge base and a similarity measure is computed between the selection rule and the extracted sub-ontology (Paragraph. 6.3.2). This measure indicates if some knowledge in the knowledge base is semantically close to the rule description. (2) Then, for each concept defined in the extracted sub-ontology, we extract the associated instances and rank it based on the ratio ($\in \mathbb{R}$) between the amount of terms used to define the instance that are common with terms defined in the selection rule over the total amount of terms in the selection rule (Paragraph. 6.3.3). Weighted by the similarity measure, it gives a relevancy measurement ($\in \mathbb{R}$) for each returned instances (actually it gives a confidence level about the distance between the returned instances and the selection rule intrinsic semantics).

Since the selection rules are defined at design time and therefore cannot be modified nor tuned at run time, their content is enriched at over time (and their expressivity increased) by merging new knowledge (brought by newly discovered devices). This enrichment helps keeping the similarity measure as high as possible over time (note that this approach suffers from the content size and validity issues exposed in chapter 4 and for which some solutions are given in chapter 5).

Preliminary experiment results have been described in Paragraph. 6.4 but still need to be enriched with additional datasets and ontology similarity algorithms not tested so far (more particularly the structural distance algorithm).

6.6 References

[1] Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation, January 2008.

[2] Harris, S., Seaborne, A., & Prud'hommeaux, E. (2013). SPARQL 1.1 query language. W3C Recommendation, 21.

[3] Glimm, B., & Krötzsch, M. (2010). SPARQL beyond subgraph matching. In The Semantic Web–ISWC 2010 (pp. 241-256). Springer Berlin Heidelberg.

[4] Pérez, J., Arenas, M., & Gutierrez, C. (2006, November). Semantics and Complexity of SPARQL. In International semantic web conference (Vol. 4273, pp. 30-43).

[5] Delpeuch, A., & Preller, A. (2014, April). From natural language to RDF graphs with pregroups. In EACL'2014: 14th Conference of the European Chapter of the Association for Computational Linguistics (pp. 55-62). EACL.

[6] Draicchio, F., Gangemi, A., Presutti, V., & Nuzzolese, A. G. (2013). Fred: From natural language text to RDF and owl in one click. In The Semantic Web: ESWC 2013 Satellite Events (pp. 263-267). Springer Berlin Heidelberg.

[7] Tulasi, R. L., & Rao, M. S. (2014). Survey on Techniques for Ontology Interoperability in Semantic Web. Global Journal of Computer Science and Technology, 14(2).

[8] Ruotsalo, T., & Hyvönen, E. (2007, January). A method for determining ontology-based semantic relevance. In Database and Expert Systems Applications (pp. 680-688). Springer Berlin Heidelberg.

[9] Sun, Z., Wang, H., Wang, H., Shao, B., & Li, J. (2012). Efficient subgraph matching on billion node graphs. Proceedings of the VLDB Endowment, 5(9), 788-799.

[10] Zou, L., Mo, J., Chen, L., Özsu, M. T., & Zhao, D. (2011). gStore: answering SPARQL queries via subgraph matching. Proceedings of the VLDB Endowment, 4(8), 482-493.

[11] Saruladha, K., Aghila, G., & Raj, S. (2010, February). A survey of semantic similarity methods for ontology based information retrieval. In Machine Learning and Computing (ICMLC), 2010 Second International Conference on (pp. 297-301). IEEE.

[12] Mädche, A., Staab, S.: Measuring similarity between ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 251–263. Springer, Heidelberg (2002)

[13] David, J., Euzenat, J.: Comparison between ontology distances (preliminary results). In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 245–260. Springer, Heidelberg (2008)

[14] David, J., Euzenat, J., & Šváb-Zamazal, O. (2010). Ontology similarity in the alignment space. In The Semantic Web–ISWC 2010 (pp. 129-144). Springer Berlin Heidelberg.

[15] Euzenat, J., Loup, D., Touzani, M., & Valtchev, P. (2004). Ontology alignment with OLA. In Proc. 3rd ISWC2004 workshop on Evaluation of Ontology-based tools (EON) (pp. 59-68). No commercial editor.

[16] Campinas, S., Perry, T. E., Ceccarelli, D., Delbru, R., & Tummarello, G. (2012, September). Introducing RDF graph summary with application to assisted SPARQL formulation. In Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on (pp. 261-266). IEEE.

[17] Zhang, X., Cheng, G., & Qu, Y. (2007, May). Ontology summarization based on RDF sentence graph. In Proceedings of the 16th international conference on World Wide Web (pp. 707-716). ACM.

[18] Khatchadourian, S., & Consens, M. (2010). Explod: Summary-based exploration of interlinking and rdf usage in the linked open data cloud. The Semantic Web: Research and Applications, 272-287.

7. Conclusion and Future Work

7.1 Summary

Semantic web technologies are gaining interest in the WoT (Web of Things) community for their ability at managing the increasing semantic heterogeneity between devices. Thus, by qualifying the devices with semantic annotations describing the devices and the services and relying on a formal knowledge model (SWoT, Semantic Web of Things), ambient systems have now the ability to understand and reason about it hereby leading to select, among all the available devices and services, the most relevant ones.

In the **chapter 2 (Knowledge Modeling),** we have reviewed the main technologies used in the domain of the semantic web to formally describe the knowledge (Paragraph. 2.2). We have described the underlying theoretical foundations of these technologies and addressed it from two points of views: (1) The knowledge expressivity they provide, (2) their computational complexity. These two aspects are key to ensure devices and services selection relevancy and keep the overall system responsiveness at an acceptable level. Then we have given an overview (Paragraph. 2.3) of the knowledge types to be described in the context of ambient environments (namely the contextual, the structural and the functional knowledge types). Such a variety of knowledge domains to cover enforces the need for a high knowledge model expressivity. While expressivity is key at reducing the knowledge abstraction level and then helps ambient systems to better segregate devices and services, it implies computational complexity that may be problematic for ambient systems with low computational resources. We concluded (Paragraph 2.4) that OWL (2) DL are the knowledge description languages better fitting with the expressivity/computational complexity constraints. While most of the approaches rely on specific and static knowledge models to qualify the devices, we have proposed a knowledge modeling approach based on heterogeneous devices and services description models (Multiple Local Ontologies, MLO) whose foundation rely on the fact that with regards to the numerous IoT manufacturers, it is unlikely that a commonly agreed and comprehensive world knowledge description model will be developed and used (Paragraph. 2.6).

Providing the manufacturers with tools helping them to rapidly extract and encapsulate the knowledge into metadata is key to reduce the products design to market cycle time. In the **chapter 3 (Knowledge Extraction)**, we have first investigated how to turn already existing formal knowledge into metadata either from upper ontology, from which only the statements needed to qualify a given concept are extracted, or from web pages whose data content is published as interlinked RDF graphs (Linked Data) whose meta models rely on Linked Open Vocabularies (LOV). Then we have reviewed the several technologies used to convey the knowledge from the devices to the ambient systems (web service description languages) and challenge them with regards to their ability at conveying all the necessary knowledge types to be described (Paragraph. 3.3). Two approaches can be used depending on the situation: (1) use a top down approach where an upper ontology is used to describe the web services functionalities, (2) use a bottom up approach extending existing web services description languages (mainly WSDL) with semantics capabilities. These description languages are mainly dedicated at defining the semantics of the services functionalities (input, output, execution flow, etc…) but not the non-functional knowledge (structural or contextual) other than the Quality of Service (QoS). We have proposed to mix-up the MLO/MLO-LD approach with the bottom-up approach (assuming that manufacturers will not rely on an upper ontology to describe the functionalities of the services).

In the **chapter 4 (Knowledge Integration)**, we have first described the knowledge base foundations (Paragraph. 4.2) and the usefulness of reasoners to infer implicit facts (Paragraph. 4.3). Knowledge bases are at the heart of the ambient systems while the knowledge they contain (assertions (*ABox*) and terminology (*TBox* or General Concept Inclusion(GCI))) is brought by the devices plunged in physical environments, theater of physical phenomena implying a dynamic knowledge integration management at the ABox and the GCI levels (Paragraph. 4.4. While, heterogeneous knowledge models imply some integration challenges to be overcame (Paragraph 4.5, concepts and terms alignments whose results depend on a hardcoded threshold value), it gives outstanding perspectives in term of services selection relevancy improvement. Indeed, as the same terms might be differently described in heterogeneous ontologies, they may acquire a more accurate meaning throughout an evolutionary process, their meaning (GCI) being shaped through the assertions done by their publishers (The IoT manufacturers). Then, an overall integration model is presented (Paragraph. 4.7) that makes GCI persistent in the knowledge base. It has been validated on two motivating scenarios

(Paragraph. 4.8). The comparison of the overall approach with previous approaches is summarized in the Figure 51:

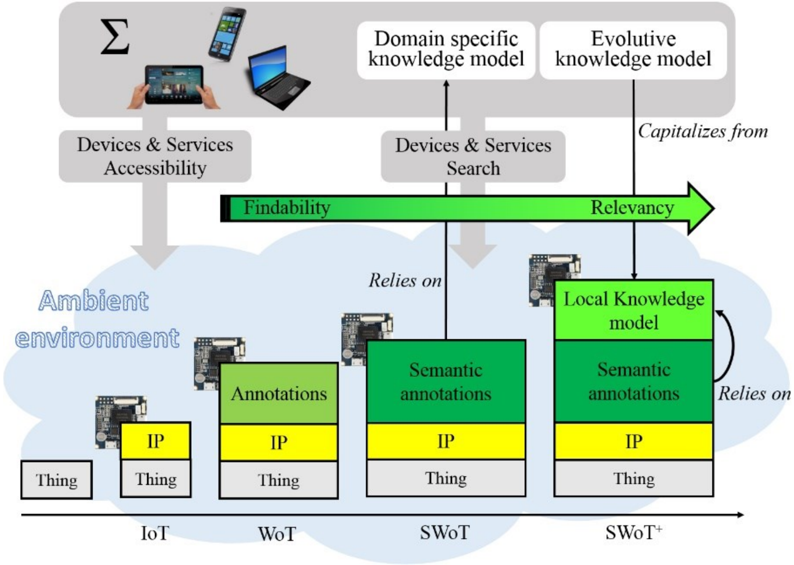


Figure 51 : From IoT to SWOT+

Since the terminological knowledge is continuously enriched over time, we have reviewed, in the **chapter 5 (Knowledge Management)** some knowledge base management solutions to address: (1) the knowledge base terminological content containment (Paragraph. 5.2), and (2) its quality by ensuring content consistency, coherency and validity (freshness) (Paragraph. 5.3).

Finally in the **chapter 6 (Knowledge Retrieval)**, we have given an overview of SPARQL (Paragraph. 6.2), the mainly used knowledge base query language, along with its foundations. We have demonstrated that the underlying knowledge retrieval model (subgraph matching) is not suitable when used to define expressive selection rules on unknown content (indeed, from the ambient system standpoint, the knowledge being enriched over time, it is unknown at design time when selection rules are defined). So, we have proposed an approach where the selection rules are expressed semantically from heterogeneous knowledge models (Paragraph. 6.3). A subgraph, similar to the selection rule description model, is then tentatively extracted from the knowledge base (with a similarity confidence level). Based on this approach, we have proposed a model to rank the instances associated to the extracted subgraph (the selection rule results) which goal is to give an insight about the semantic distance in between the selection rule intrinsic meaning and the returned results (selection relevancy measure). To keep rules expressivity as high as possible over time and then keep similarity measure and ranking as accurate as possible, incoming knowledge brought by the devices over time is merged with the rules. The approach has been validated on a use-case (Paragraph. 6.4)

7.2 Future work

The work done in this document covers a large spectrum of domains (knowledge modeling, knowledge extraction, knowledge integration, knowledge management and knowledge retrieval) the ambient system service selection mechanism depends on to ensure the relevancy of the selected devices and services. Despite the fact that the problem statements and the potential solutions have been given for each domain, only the knowledge modeling (chapter 2), the knowledge integration (chapter 4) and the knowledge retrieval (chapter 6) domains have been investigated and validated.

Further investigations and developments would be needed:

1) The knowledge extraction domain (chapter 3) is currently addressed through a long term collaboration with ESPRIT (Tunisia). The goal is to develop tools to automatically generate reusable knowledge models from web pages, manufacturer's documentations, etc…, relying on linked data (LD) and linked open vocabularies (LOV).
2) Detecting and fixing knowledge base incoherencies and inconsistencies (chapter 5) is key to ensure selected devices and services relevancy over time. Moreover, fixing such issues (or put in place

solutions to avoid it) without jeopardizing the meaning of the terms described in the knowledge base is also important and may lead to the development of novel algorithms.

3) Regarding the knowledge retrieval domain (chapter 6), the proposed approach is promising to give the system a metrics on the relevancy of the selected instances with regards to the selection rule. But additional investigations would be needed, in particular, on the usage of: (1) structural measures (rather than purely lexical measures) and (2) summary graphs that would potentially improve similarity measurements.
4) An important parameter for ambient systems is their responsiveness. We have discussed several algorithms in the chapter 4 about reasoning, integration, alignments and in the chapter 6 about sub-ontology extraction, similarity measurements, etc… So far, we do not have any clear picture about their impacts on the overall ambient system responsiveness.

While structural and contextual knowledge are key in the process of selecting the most relevant devices and services available in the environment, the functional knowledge is key at ensuring the selected services functional interoperability. As we have seen in the chapter 3, several standards exists to semantically describe services functionality through web service description languages and are classified in two approaches : (1) the top-down approach where web service descriptions are based on upper ontologies, (2) the bottom-up incremental approach enriching existing web services description standards (WSDL) with additional extensions to connect the syntactic definitions to their semantic annotations. Since: (1) it is unlikely that manufacturers will rely on an upper ontology to describe their products (including their functionalities), and (2) we want a fine grained services description model expressivity, the bottom-up approach seems to be the one to adopt. It would be interesting to investigate, how this approach, beyond its capabilities at ensuring interoperability between services, can, a step further, be used for the system to deduce the semantic of an overall composition (Figure 52)

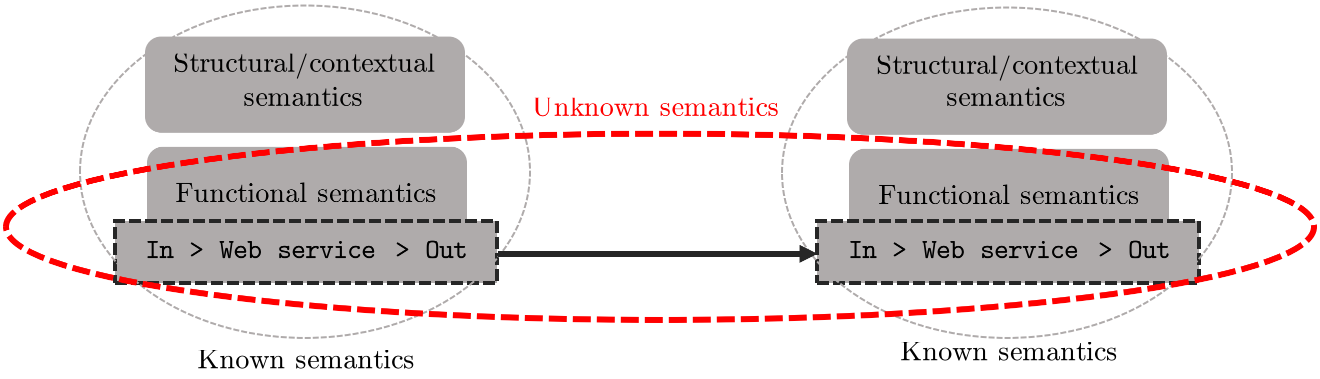


Figure 52 : From functional composition to semantic composition

8. Bibliography

[1] Haller, S. (2010). The things in the internet of things. Poster at the (IoT 2010). Tokyo, Japan.

[2] Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., & Schreier, G. (2010, April). The internet of things for ambient assisted living. In Information Technology: New Generations (ITNG), 2010 Seventh International Conference on (pp. 804-809). Ieee.

[3] Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805.

[4] Zeng, D., Guo, S., & Cheng, Z. (2011). The web of things: A survey. Journal of Communications, 6(6), 424-438.

[5] Cheung, D., Tigli, J. Y., Lavirotte, S., & Riveill, M. (2006, June). Wcomp: a multi-design approach for prototyping applications using heterogeneous resources. In Rapid System Prototyping, 2006. Seventeenth IEEE International Workshop on (pp. 119-125), IEEE.

[6] Hourdin, V., Tigli, J. Y., Lavirotte, S., Rey, G., & Riveill, M. (2008, September). SLCA, composite services for ubiquitous computing. In Proceedings of the International Conference on Mobile Technology, Applications, and Systems (p. 11). ACM.

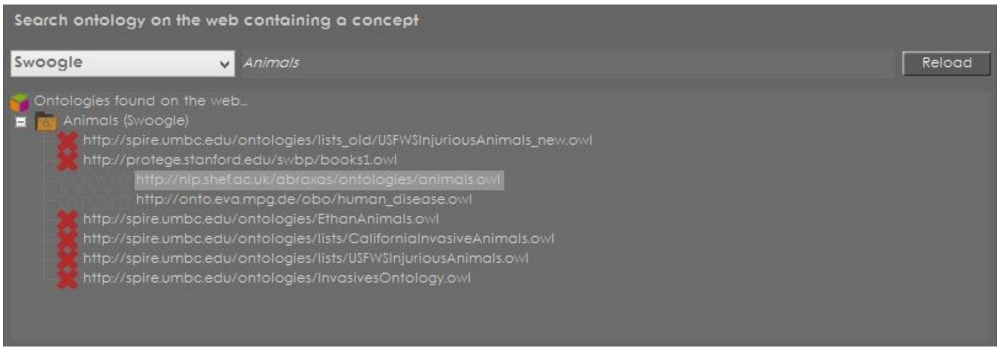
9. Annexes

9.1 Sub-Ontology Extraction Tool

9.1.1 Introduction

For the purpose of validating the approaches presented in this document and helping to generate some dataset, we have developed a small tool (still under development), whose main functionalities are explained in the next paragraphs. The tool has been developed with Java and relies on Jena API [1] (knowledge base and SPARQL endpoint), Alignment API [2] (ontology alignment and matching), OntoSim [2] (ontology similarity measurement) and WebVOWL [3] for ontology graphical visualization. WordNet [4] and LOV[1] are made available to enrich ontology with equivalent classes.

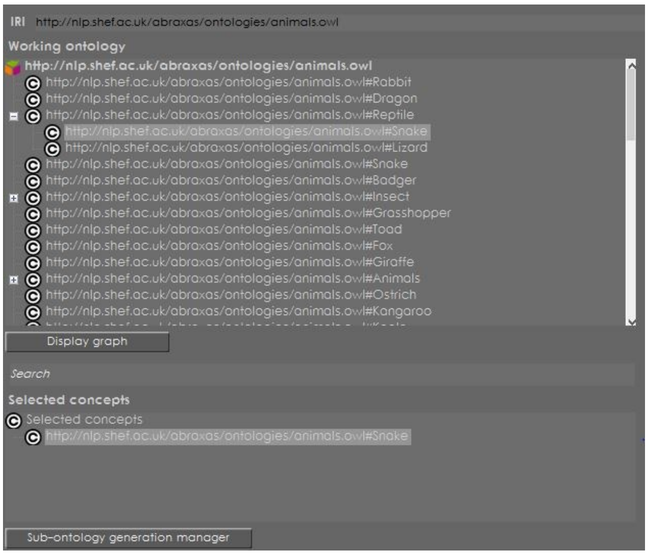
9.1.2 Gathering Ontology from the Web



Ontologies found on the web...
Animals (Swoogle)
http://spire.umbc.edu/ontologies/lists_old/USFWSInjuriousAnimals_new.owl
http://protege.stanford.edu/swbp/books1.owl
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl
http://onto.eva.mpg.de/obo/human_disease.owl
http://spire.umbc.edu/ontologies/EthanAnimals.owl
http://spire.umbc.edu/ontologies/lists/CalifornialnvasiveAnimals.owl
http://spire.umbc.edu/ontologies/lists/USFWSInjuriousAnimals.owl
http://spire.umbc.edu/ontologies/InvasivesOntology.owl

Figure 53 : Gathering ontology from the web

The tool can find from the web (thanks to Swoogle web service [5] for the time being, Watson Semantic web search [5] is in the pipe), ontologies containing a specify keyword. In the example in the Figure 53, ontologies containing the keyword '`Animals`' are retrieved. The tool check out for their accessibility. In the rest of this chapter, examples are taken from http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl. The selected ontology content can be visualized with WebVOWL [3] (Figure 53).



IRI http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Rabbit
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Dragon
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Reptile
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Lizard
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Badger
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Insect
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Grasshopper
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Toad
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Fox
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Giraffe
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Animals
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Ostrich
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Kangaroo

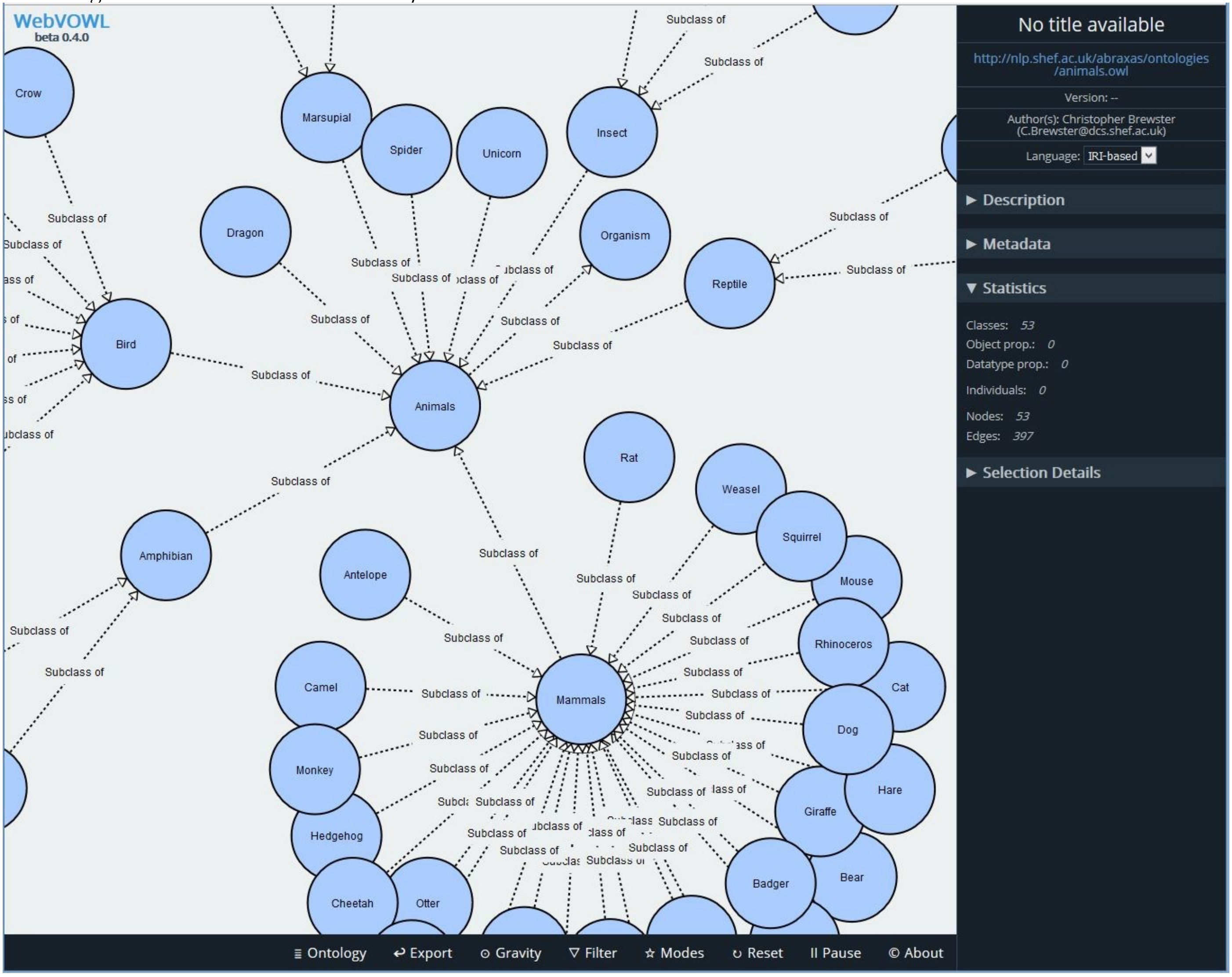
Selected concepts
http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake

9.1.3 Extracting Sub-Ontology

Once the ontology to work with has been selected, all defined classes are displayed. The user select one or more classes to work with. “Snake” concept is chosen (Figure 54).

[1] http://lov.okfn.org/dataset/lov/

Figure 54 : Selection of the concept to be extracted in a



No title available

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

Version: --

Author(s): Christopher Brewster (C.Brewster@dcs.shef.ac.uk)

Language: IRI-based

▶ Description

▶ Metadata

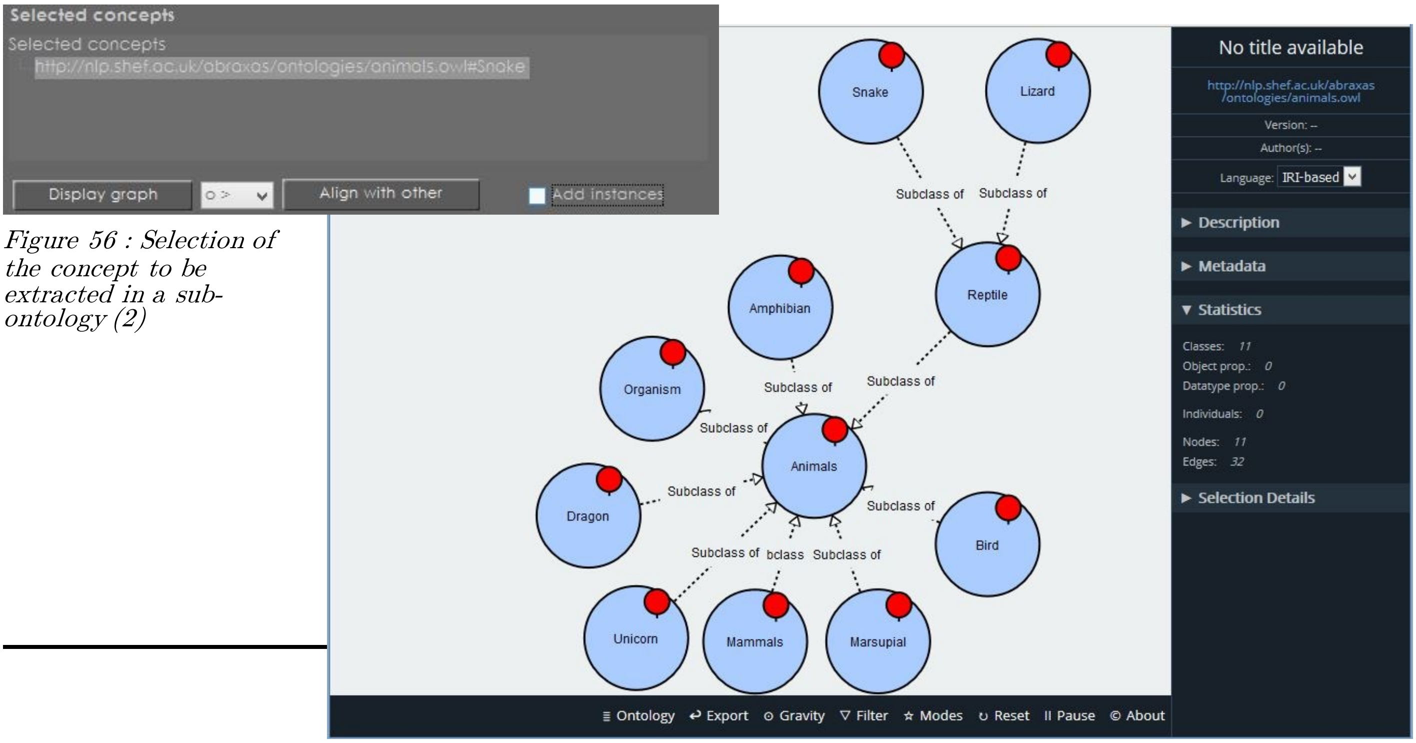
▼ Statistics

Classes: 53
Object prop.: 0
Datatype prop.: 0
Individuals: 0
Nodes: 53
Edges: 397

▶ Selection Details

Figure 55 : Ontology vizualization with WebVOWL

The Sub-ontology generation manager opened (Figure 56), the selected concept associated sub-ontology can be visualized (Figure 57).


Selected concepts

Selected concepts

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake

No title available

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

Version: --

Author(s): --

Language: IRI-based

▶ Description

▶ Metadata

▼ Statistics

Classes: 11
Object prop.: 0
Datatype prop.: 0
Individuals: 0
Nodes: 11
Edges: 32

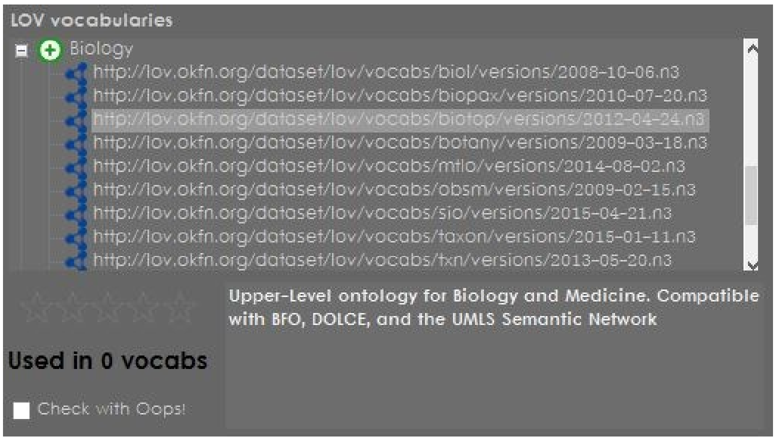
▶ Selection Details

Figure 56 : Selection of the concept to be extracted in a sub-ontology (2)

Figure 57 : Snake sub-ontology

a) Aligning Concepts with LOV and WordNet

Once the sub-ontology is generated the tool allows it to be enriched with LOV equivalent classes. The process is manual: (1) the user select the vocabulary to be used for the alignment (Figure 58), (2) the alignment algorithm to be used [2] and (3) the threshold value and finally (4) select the terms to be aligned based a equivalence strength value (Figure 59).



LOV vocabularies

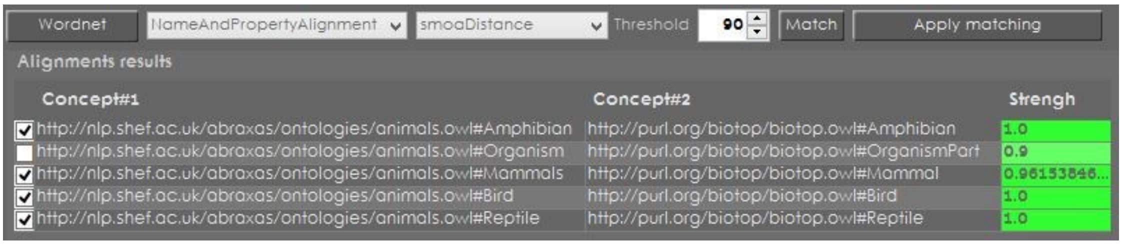
Biology
- http://lov.okfn.org/dataset/lov/vocabs/biol/versions/2008-10-06.n3
- http://lov.okfn.org/dataset/lov/vocabs/biopax/versions/2010-07-20.n3
- http://lov.okfn.org/dataset/lov/vocabs/biotop/versions/2012-04-24.n3
- http://lov.okfn.org/dataset/lov/vocabs/botany/versions/2009-03-18.n3
- http://lov.okfn.org/dataset/lov/vocabs/mtlo/versions/2014-08-02.n3
- http://lov.okfn.org/dataset/lov/vocabs/obsm/versions/2009-02-15.n3
- http://lov.okfn.org/dataset/lov/vocabs/sio/versions/2015-04-21.n3
- http://lov.okfn.org/dataset/lov/vocabs/taxon/versions/2015-01-11.n3
- http://lov.okfn.org/dataset/lov/vocabs/txn/versions/2013-05-20.n3

Upper-Level ontology for Biology and Medicine. Compatible with BFO, DOLCE, and the UMLS Semantic Network

Used in 0 vocabs

□ Check with Oops!

Figure 58 : LOV vocabulary selection



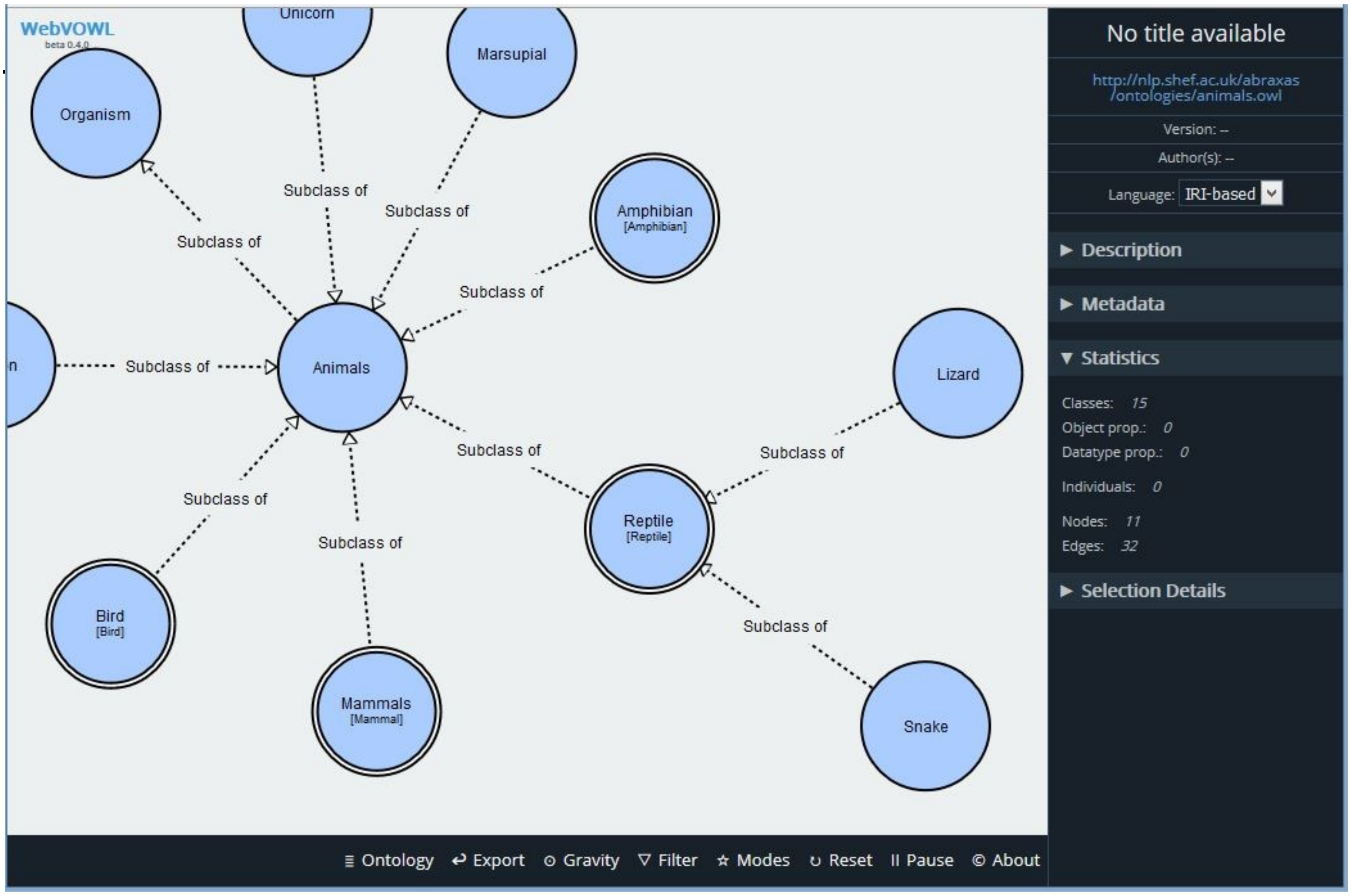
Wordnet | NameAndPropertyAlignment | smoaDistance | Threshold 90 | Match | Apply matching

Alignments results

	Concept#1	Concept#2	Strengh
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Amphibian	http://purl.org/biotop/biotop.owl#Amphibian	1.0
□	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Organism	http://purl.org/biotop/biotop.owl#OrganismPart	0.9
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Mammals	http://purl.org/biotop/biotop.owl#Mammal	0.96153846...
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Bird	http://purl.org/biotop/biotop.owl#Bird	1.0
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Reptile	http://purl.org/biotop/biotop.owl#Reptile	1.0

Figure 59 : Alignment algorithm and threshold value selection

The alignment is then applied (Apply Matching) and the resulting enriched sub-ontology can be visualized (Figure 60).



No title available

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

Version: --

Author(s): --

Language: IRI-based

▶ Description

▶ Metadata

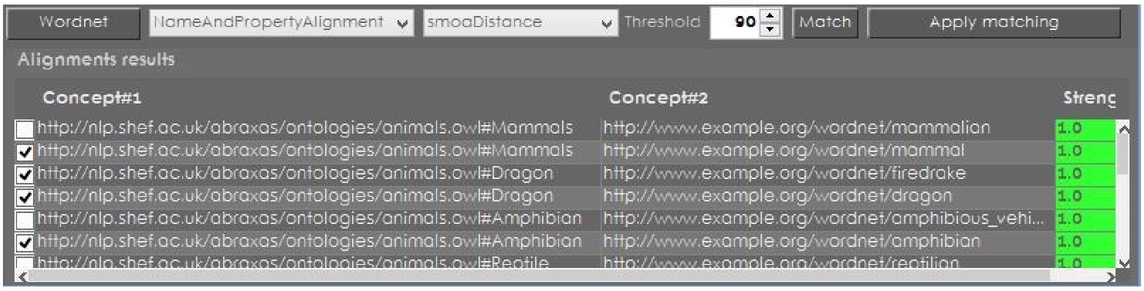
▼ Statistics

Classes: 15
Object prop.: 0
Datatype prop.: 0
Individuals: 0
Nodes: 11
Edges: 32

▶ Selection Details

Figure 60 : Snake sub-ontology aligned with LOV vocabulary

The same approach is used to align the sub-ontology with WordNet synonyms (Figure 61)

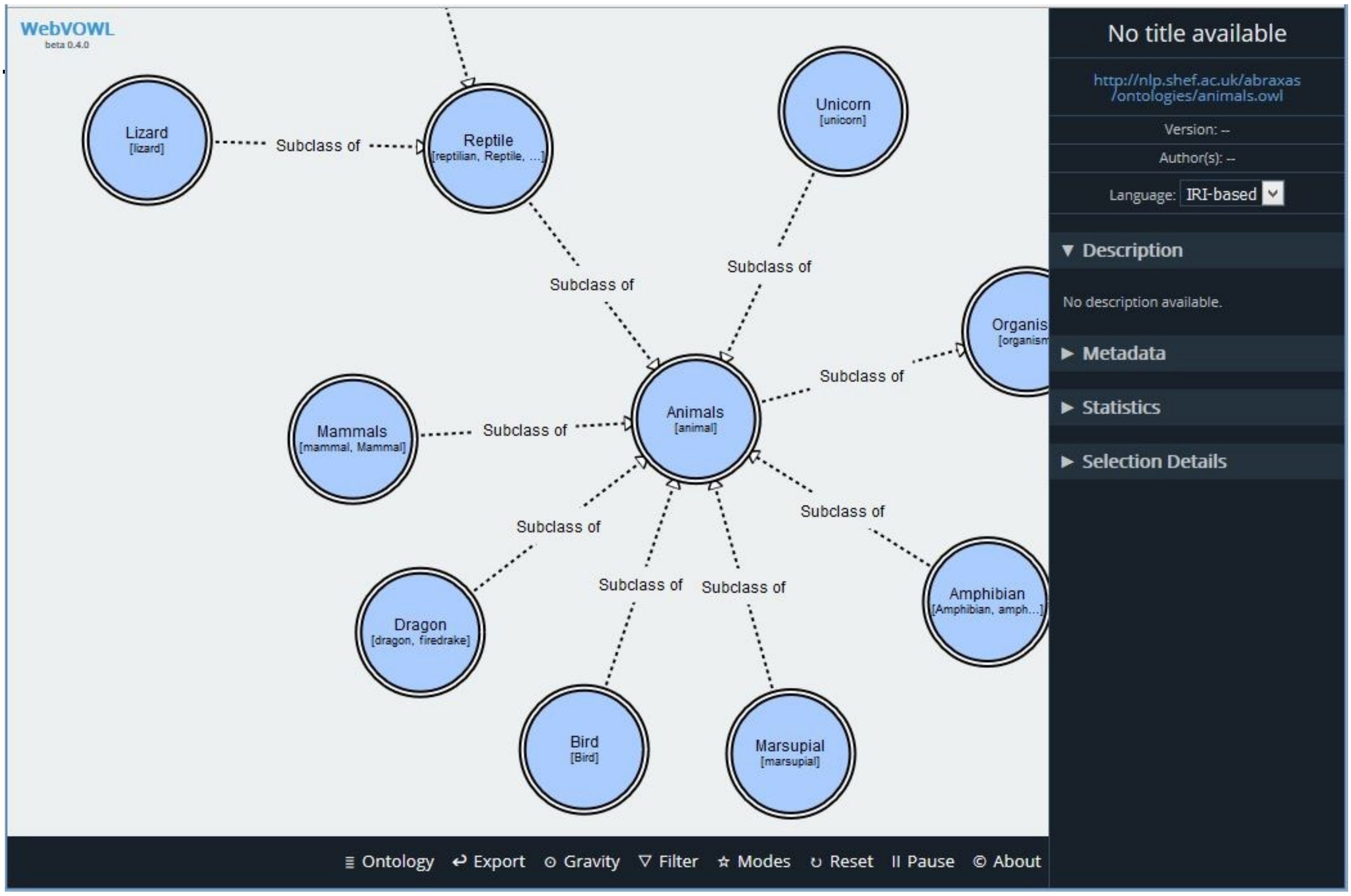


Threshold 90

Alignments results

	Concept#1	Concept#2	Streng
□	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Mammals	http://www.example.org/wordnet/mammalian	1.0
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Mammals	http://www.example.org/wordnet/mammal	1.0
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Dragon	http://www.example.org/wordnet/firedrake	1.0
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Dragon	http://www.example.org/wordnet/dragon	1.0
□	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Amphibian	http://www.example.org/wordnet/amphibious_vehi...	1.0
☑	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Amphibian	http://www.example.org/wordnet/amphibian	1.0
□	http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Reptile	http://www.example.org/wordnet/reptilian	1.0

Figure 61 : Alignment with WordNet synonyms



No title available

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

Version: --

Author(s): --

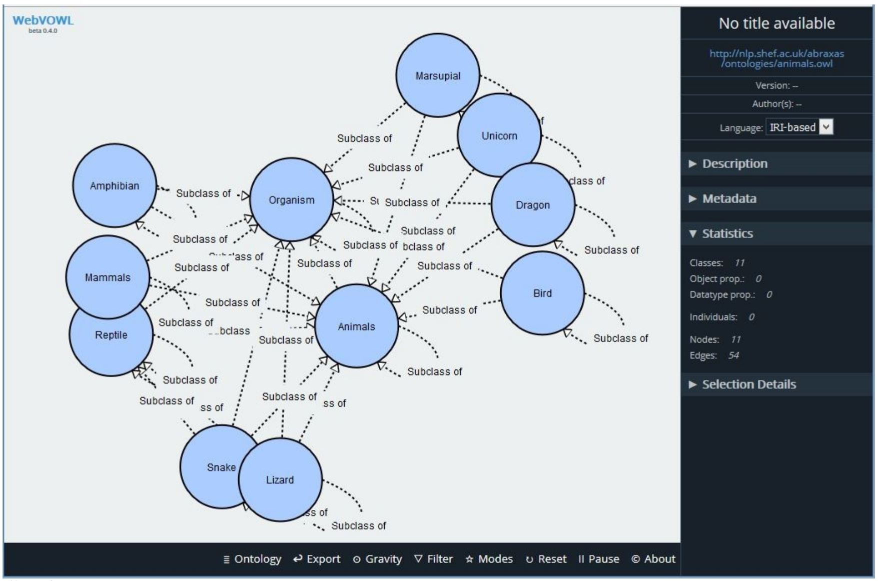
Language: IRI-based

No description available.

Figure 62 : Snake ontology aligned with WordNet synonyms

b) Reasoning on the Sub-Ontology

The tools allows to add inferences on the sub-ontology. (Figure 63) Inferences are based on the Jena API available reasoners[1]



No title available

http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl

Version: --

Author(s): --

Language: IRI-based

Classes: 11

Object prop.: 0

Datatype prop.: 0

Individuals: 0

Nodes: 11

Edges: 54

Figure 63 : Transistive inferences added to the sub-ontology

[1] https://jena.apache.org/documentation/inference/#owl

c) Degrading Sub-Ontology

For the purpose of the experiment described in Paragraph. 4.8.1, a dataset has been needed to demonstrate the knowledge base GCI enrichment over time. DogOnt ontology [6] rev 3.2.11 describing 926 concepts and containing 9383 axioms has been used for that purpose. The dataset has been created by fragmenting the ontology into sub-ontologies defining and structuring all the knowledge necessary to fully describe some concepts (Oven, Fridge, Computer, Cooker, Boiler, etc…). Then, from each sub-ontology, has been generated a set of *degraded* sub-ontologies containing a subset of the concepts complete knowledge (Figure 33).

To do so, the tool gives an overview of the estimated knowledge coverage by depth (Figure 64) or by probability to have the concepts added in the resulting sub-ontology (Figure 65). The tool also gives a status about the resulting sub-ontology consistency (Figure 66).

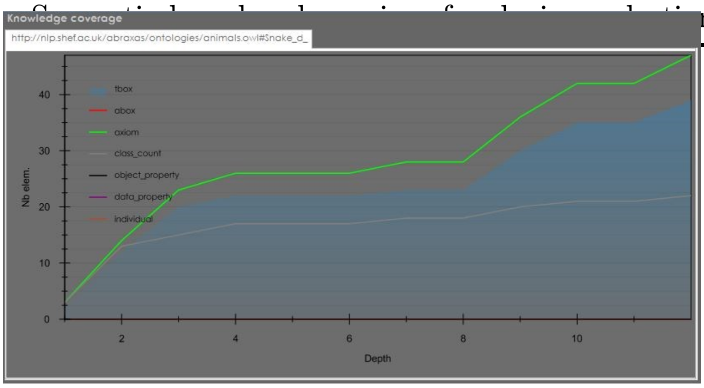


Figure 64 : Sub-ontology knowledge coverage by depth

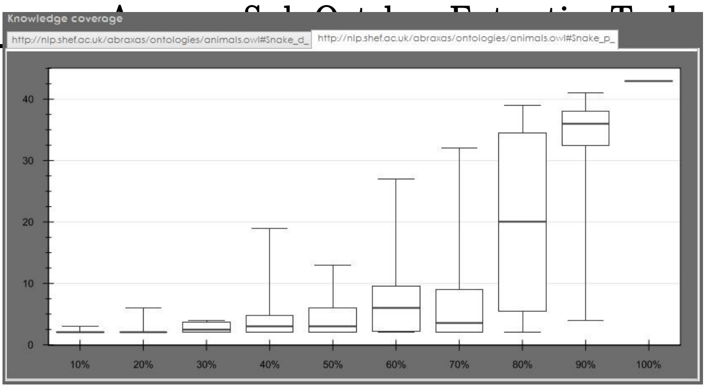
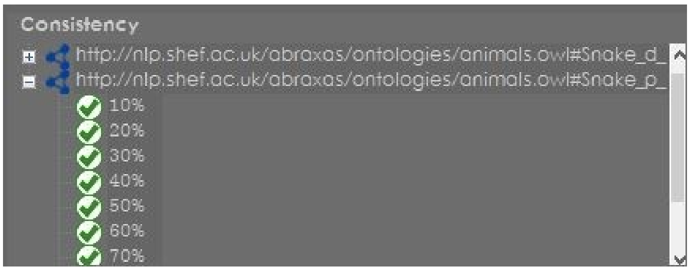


Figure 65 : Sub-ontology knowledge coverage by probability



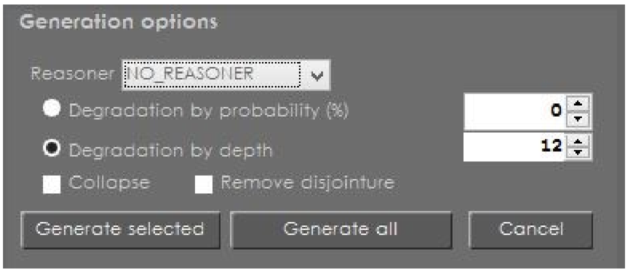
Consistency

⊞ http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake_d_
⊟ http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#Snake_p_
- ✓ 10%
- ✓ 20%
- ✓ 30%
- ✓ 40%
- ✓ 50%
- ✓ 60%
- ✓ 70%

Figure 66 : Sub-ontology consistency checking

d) Generating Sub-Ontology

Once the user has defined LOV and/or WordNet equivalences, he can execute the sub-ontology generation to a file (Figure 67). User can select the reasoner to be applied on the sub-ontology and defined the degradation rate (either depth or probability). If multiple concepts are simultaneously selected, the user can either generate a sub-ontology containing all selected concepts (Collapse) or select a sub-ontology for each of them.



Generation options

Reasoner: NO_REASONER

○ Degradation by probability (%): 0

● Degradation by depth: 12

□ Collapse □ Remove disjointure

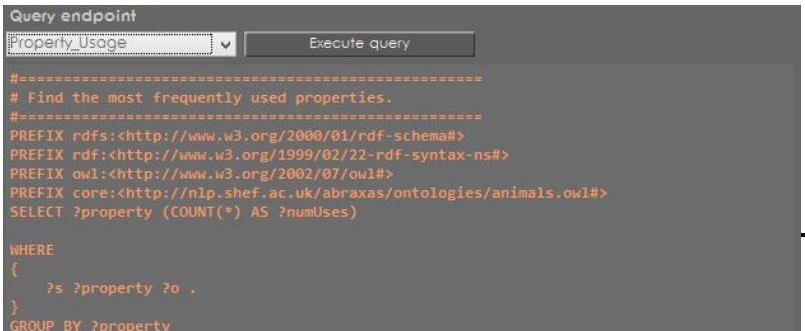
Figure 67 : Sub-ontology generation

9.1.4 Measuring the Similarity Between Ontologies

Beside aforementioned features, the tool gives the capability to measure the similarity between two ontologies. This measure is based on OntoSim API [2] and is based, for the time being only on lexical measurements. Ontology structural-based similarity measurements are on the pipe…

9.1.5 SPARQL Endpoint

A SPARQL end-point has been developed to query the knowledge base content. Several ontology structural metrics based on SPARQL queries are implemented by default [7].

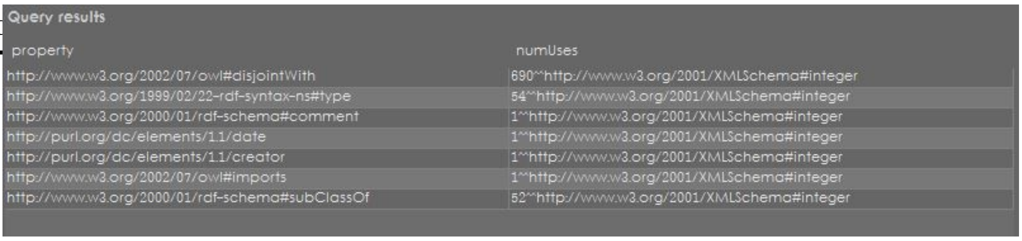


Query endpoint

Property_Usage

```
#=====================================================
# Find the most frequently used properties.
#=====================================================
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX core:<http://nlp.shef.ac.uk/abraxas/ontologies/animals.owl#>
SELECT ?property (COUNT(*) AS ?numUses)

WHERE
{
    ?s ?property ?o .
}
GROUP BY ?property
```



Query results

property	numUses
http://www.w3.org/2002/07/owl#disjointWith	690^^http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	54^^http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/2000/01/rdf-schema#comment	1^^http://www.w3.org/2001/XMLSchema#integer
http://purl.org/dc/elements/1.1/date	1^^http://www.w3.org/2001/XMLSchema#integer
http://purl.org/dc/elements/1.1/creator	1^^http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/2002/07/owl#imports	1^^http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/2000/01/rdf-schema#subClassOf	52^^http://www.w3.org/2001/XMLSchema#integer

Figure 68 : SPARQL endpoint

9.1.6 Conclusion

This very first revision of the tool is quite useful and complete for the purpose of generating sub-ontology from upper ontology. It is far from being perfect and several enhancements have to be done: (1) use sub-ontology extraction algorithms as defined in Paragraph. 3.2.1, (2) add structural-based ontology similarity measurement algorithms, (3) add Watson Semantic web search engine, (4) etc…

9.1.7 References

[1] JENA, A. (2011). Jena–A Semantic Web Framework for Java. Talis Systems.

[2] David, J., Euzenat, J., Scharffe, F., & Dos Santos, C. T. (2011). The alignment api 4.0. Semantic web journal, 2(1), 3-10.

[3] Lohmann, S., Link, V., Marbach, E., & Negru, S. (2014). WebVOWL: Web-based visualization of ontologies. In Knowledge Engineering and Knowledge Management (pp. 154-158). Springer International Publishing.

[4] Miller, G. A., Fellbaum, C., Tengi, R., Wakefield, P., LANGONE, H., & HASKELL, B. (2013). Wordnet: A lexical database for the English Language. 2002. Available from:¡ cogsci. princeton. edu/wn.

[5] Singh, G., & Jain, V. (2014). Information Retrieval (IR) through Semantic Web (SW): An Overview. arXiv preprint arXiv:1403.7162.

[6] Bonino, D., & Corno, F. (2008). Dogont-ontology modeling for intelligent domotic environments (pp. 790-803). Springer Berlin Heidelberg.

[7] Holst, T. (2013). Structural analysis of unknown RDF datasets via SPARQL endpoints (Doctoral dissertation, Master thesis defense 11).